A Text Editor

MULTI-EDIT

Ver 3.0

(C) Copyright 1987, 1988, 1989 by European Cybernetics

User's Guide
and
Reference Manual

Place
Stamp
Here

EUROPEAN CYBERNETICS
Bothaplein 6
6814 AJ ARNHEM
The Netherlands

-------------------(fold and tape)-------------------

TABLE OF CONTENTS

1. LICENSE AGREEMENT 1
    LIMITED WARRANTY 1
    TECHNICAL SUPPORT 1

2. GETTING STARTED 2
    HARDWARE REQUIREMENTS 3
    INSTALLING MULTI-EDIT ON YOUR SYSTEM 3

3. BASIC CONCEPTS 4
    DEFAULT FUNCTION KEY ASSIGNMENTS 6

4. CURSOR CONTROL 7
    MOUSE SUPPORT 9
    POSITION MARKING 9

5. INSTALL AND SETUP OPTIONS 10
    SETUP OPTIONS 10
    THE EDIT SETTINGS MENU 11
    THE SCREEN SETUP MENU 12
    THE COLOR SELECTION MENU 12
    KEY MAPPING 13
    THE MOUSE MENU 14
    FILENAME EXTENSION SETUP 15
    THE PRINTER TYPE MENU 15

6. COMMAND LINE OPTIONS 16

7. LOADING AND SAVING FILES 18

8. INSERTING TEXT 20

9. DELETING TEXT 20

10. TABS 21

11. BLOCK OPERATIONS 22
    BLOCK MARKING COMMANDS 23
    BLOCK MANIPULATION 24
    BLOCK INDENTING 25

12. UNDO 26

13. WINDOWS 27
    THE SWITCH WINDOW MENU 29
    SWITCH WINDOW MENU COMMANDS 29

14. FILE FORMATTING 31

15. FORMAT LINES 33

16. PAGE BREAKS 34

17. PRINT FORMATTER 35
    FORMATTING CODES 36
    PAGINATION, HEADERS, AND FOOTERS 36
    COMMENT AND IGNORED LINES 36
    LINE SPACING, MARGINS, AND CENTERING 37
    PRINT ATTRIBUTES 37
    USER DEFINABLE FORMATTING COMMANDS 38
    TABLE OF CONTENTS GENERATOR 39

18. PRINTING AND PRINTER CONTROL 41

19. SEARCH AND REPLACE 43
    SEARCH OPTIONS 44
    REGULAR EXPRESSIONS 45
    REPLACE STRING EXPRESSIONS 45
    SEARCH AND REPLACE EXAMPLES 46

20. DOS DIRECTORY SHELL 47

21. RUNNING DOS FROM MULTI-EDIT 50

22. KEYSTROKE MACROS 51
    EDITING KEYSTROKE MACROS 51
    SAVING AND RELOADING KEYSTROKE MACROS 52

23. LINE DRAWING 53

24. CALCULATOR 54

25. ASCII TABLE 55

26. CONDENSED DISPLAY 55

27. FILENAME EXTENSIONS 56

28. INDENTING STYLES 59

29. LANGUAGE SPECIFIC FEATURES 60
    SMART AUTO-INDENT 60
    AUTOMATIC LINE NUMBERING FOR BASIC 60
    BRACE, BRACKET AND STRUCTURE MATCHING 61
    TEMPLATE EDITING 62
    COMPILING FROM MULTI-EDIT 64
    NEXT ERROR 64
    COMPILER COMMAND LINES 65
    COMPILER PROMPT 65
    REDIRECTING OUTPUT IN BATCH FILES 66
    LARGE COMPILE 67

30. MACRO LANGUAGE INTERFACE 68

31. MISCELLANEOUS STUFF 69
    COMMAND REPEAT 69
    TIME AND DATE STAMPING 69
    EDITING THE HELP FILE 69

APPENDIX A: EDITING FILES LARGER THAN MEMORY 70

APPENDIX B: DESQ-VIEW, WINDOWS and OS/2 70

APPENDIX C: RUNNING WITH OLDER VERSIONS OF DOS 71

INDEX 73

1. LICENSE AGREEMENT

Multi-Edit is protected under United States as well as Dutch copyright law. Copies may be made for the purpose of backup ONLY. This software may be moved from computer to computer, and may be used by more than one person, as long as there is no chance of it being used at one location while it is being used at another or that it may be used by more than one person at the same time.

LIMITED WARRANTY

European Cybernetics warrants the disks in this package against defects due to materials or workmanship for a period of 30 days. If, in this time period, the disks become defective, they may be returned to us for free replacement.

European Cybernetics makes no representations of warranties as to merchantability or fitness to any particular purpose. There is no other warranty expressed or implied.

TECHNICAL SUPPORT

The *European Cybernetics User Support Bulletin Board System* may be accessed via modem at (+31) **85459034**. It provides a wealth of Multi-Edit macros and technical information. In addition the BBB provides a forum for user comments and questions. Your questions will be promptly answered by either the authors, or other users.

If your questions are more immediate, or you don't have access to a modem, please feel free to call us direct at (+31) **85427757**.

EUROPEAN CYBERNETICS
Bothaplein 6
6814 AJ ARNHEM
The Netherlands
(+31) 85427757

2. *GETTING STARTED*

Congratulations! You have in your possession Multi-Edit, the most advanced programmer's editor available today.

Multi-Edit is a completely programmable, reconfigurable, window oriented text editor. It can edit up to 100 files at once, each of which may be up to 2 gigabytes in size.

Multi-Edit's features include:

- On-line HELP for all operations.
- Access to operations from both the function keys and an easy to use menu system.
- Full UNDO of almost any editing operation, up to the last 100 changes.
- Line, column, and stream of text oriented block operations.
- Regular expression search and replace.
- Full windowing. View multiple files, or the same file in up to 100 multiple windows.
- Compile from within Multi-Edit. Interfaces to most common compilers to automatically find compiler errors.
- Multi-Edit is fully programmable and reconfigurable! Multi-Edit's macro language is both powerful and very easy to read.
- Keystroke Macros which are easy to create, edit, and save. As many as you want.

PLUS MUCH MORE!

HARDWARE REQUIREMENTS

IBM PC, XT, AT, or close compatible. MS-DOS or PC-DOS version 2.0 or later. Multi-Edit requires 200K of free RAM (not including DOS) to run. More memory is recommended for better performance. Multi-Edit can be run from a floppy disk, but it is recommended that you have a hard disk.

Multi-Edit is compatible with most mice, including Microsoft, Mouse Systems and Logitech mice.

INSTALLING MULTI-EDIT ON YOUR SYSTEM

Multi-Edit is not copy protected. Simply copy all the files on the disk to wherever you wish the program to reside. Placing it in its own directory and putting this directory in the DOS PATH is recommended (we suggest \ME).

Multi-Edit is executed by simply typing ME at the DOS command prompt.

It is also recommended that you make at least one backup on floppy of Multi-Edit and record the serial number on the disk label. This number is required for technical support.

3. BASIC CONCEPTS

Multi-Edit is a multiple window, fully programmable and reconfigurable editing environment. Multi-Edit is a very complex program, and it will take a while to learn all of its features. However, a solid grasp on the more important concepts will ease the learning path.

WINDOWS AND FILES

Multi-Edit may be used to edit up to 100 files at once. Normally each file has a window associated with it. It is NOT necessary for all of the windows to fit on the screen. Windows may overlap, and be completely hidden by other windows.

You may cycle back and forth between windows with the <ShftF1> and <CtrlF1> keys. The <AltF1> key will bring up the **Switch Window** menu, which is a complete list of all of the windows and their associated files.

Obviously it would be very easy to get things very cluttered with a lot of windows. To handle this, Multi-Edit allows you to hide and unhide windows. A hidden window will be skipped when switching back and forth between windows.

It is often desirable to view a single file in more than one window. Multi-Edit allows windows to be **Linked** together, so that any number of windows may view the same file.

Windows may also be resized, and moved with the **Modify Window** operation, available from the **Window** menu.

Files may be up to 2 gigabytes, and contain more than 2 billion lines of text. Multi-Edit will inform you if a file is too large to fit in memory. Disk swapping will be performed automatically, without the need for user intervention.

WINDOW DISPLAY

Each window displays the line and column number (and page number if in document mode) in the upper left corner of the window border. The filename will be displayed in the lower right or upper right corner of the window border, depending on the configuration. The window ID will be displayed in the lower left corner.

FILENAME EXTENSION SPECIFIC SETUP

Multi-Edit sets up tabstops, margins, language-specific information, etc., based on the filename extension of the file being edited. The setups for each filename extension are created in the **Install-Filename Extensions** menus. Each setup is completely reconfigurable by the user. Setups may be added, changed and deleted as needed. If a file is edited that has an extension not on the list of extension setups, then the **Default** setup is used. The default setup is also completely reconfigurable.

EASY CONFIGURATION

Multi-Edit does not have a separate setup program. All re-configuration is done within the editor itself, from the **Install** menu. Almost all of Multi-Edits options, including the screen layout and keyboard layout, can be changed at anytime.

UNDO

Multi-Edit supports full undo of any editing operation that changes text or make a major change to the cursor position. From 1 to 64767 operations will be stored for each file, depending on how Multi-Edit is configured.

THE STATUS LINE

The **status line** is at the top of the screen. It always contains the Mult-Edit version, the module name (Text Edit, DOS Shell, Calculator etc.), insert or overwrite mode, caps-lock status, memory status, and the current time and date. Also, if a keystroke-macro is being recorded, an 'R' will appear next to the memory status. The status line may be turned off from the **Install** menu in order to free up another line for text.

USING MENUS

Most operations may be accessed via the menu system. The main menu is assigned by default to the <F2> key. Menu selections may be made as follows:

A. Use the LEFT and RIGHT or UP and DOWN Arrow keys, or the mouse to move the selection bar to the desired menu item. Then hit ENTER or the left mouse button to select that item.

OR...

B. Simply press the highlighted capital letter of the selection.

Use the <ESC> key or the right mouse button to exit a menu without making a selection. Hitting <F1> will bring up help on the highlighted item.

THE FUNCTION KEYS

The function keys are labeled at the bottom of the screen. Notice that when the Shift, Ctrl or Alt keys are held down, the function key labels change to show the labels for that key combination.

The following is a list of the default function key assignments. They can be reprogrammed from the Keymapping section of the Install menu.

DEFAULT FUNCTION KEY ASSIGNMENTS

<F1>	Help	**<ShftF1>**	Next Window
<F2>	Main Menu	**<ShftF2>**	Stamp Date/Time
<F3>	Save file	**<ShftF3>**	Load File
<F4>	Indent	**<ShftF4>**	Undent
<F5>	Marks position	**<ShftF5>**	Goto Mark
<F6>	Search	**<ShftF6>**	Search and Replace
<F7>	Begin/End block	**<ShftF7>**	Begin Column Block
<F8>	Run a Macro	**<ShftF8>**	Delete line
<F9>	Copy Block	**<ShftF9>**	Inter-window block copy
<F10>	Move Block	**<ShftF10>**	Inter-window block move
<CtrlF1>	Previous Window	**<AltF1>**	Switch Window menu
<CtrlF2>	Next compiler error	**<AltF2>**	Calculator
<CtrlF3>	Save Block	**<AltF3>**	Shell to DOS
<CtrlF4>	DOS Directory Shell	**<AltF4>**	Condense display
<CtrlF5>	Zoom Window	**<AltF5>**	ASCII Table
<CtrlF6>	Search Again	**<AltF6>**	Match brace
<CtrlF7>	Undo	**<AltF7>**	Begin Stream Block
<CtrlF8>	Compile	**<AltF8>**	Goto Line
<CtrlF9>	Block Off	**<AltF9>**	Template
<CtrlF10>	Delete Block	**<AltF10>**	Record Keystroke macro

4. CURSOR CONTROL

The cursor may be moved to any position on the screen, regardless of whether or not the cursor is beyond the end of a line or the end of a file.

The following is a list of the default cursor control key assignments. They may be recapped as needed via the INSTALLATION AND SETUP menu. (See INSTALLATION AND SETUP)

<LEFT>	Cursor left one character
<RIGHT>	Cursor right one character.
<UP>	Cursor up one line.
<DOWN>	Cursor down one line.
<PgUp>	Cursor up one window length.
<PgDn>	Cursor down one window length.

<HOME> If <HOME> is hit once, then:

  If the cursor is beyond the current indent level, then the cursor will be move to the current indent level, otherwise it will be moved to column 1.

If <HOME> is hit twice, in rapid succession, then:

  If the cursor is NOT at the top of the window, then the cursor will go to the top of the window.

  If the cursor is at the top of the window, then the cursor will go to the top of the file.

If <HOME> is hit three times, in rapid succession, then:

  The cursor will go to the top of the file.

<END> If <END> is hit once, then:

  Then cursor will go to the end of the line.

If <END> is hit twice, in rapid succession, then:

  If the cursor is NOT at the bottom of the window, then the cursor will go to the bottom of the window.

  If the cursor IS at the bottom of the window, then the cursor will go to the end of the file.

If <END> is hit three times, in rapid succession, then:

  The cursor will go to the end of the file.

<CtrlLEFT>	Cursor one word left.
<CtrlRIGHT>	Cursor one word right.
<CtrlPgUp>	Cursor to previous page break string (see PAGE BREAKS).
<CtrlPgDn>	Cursor to next page break string (see PAGE BREAK)
<CtrlHOME>	Cursor to top of file.
<CtrlEND>	Cursor to end of file.

<CtrlU>	Scroll window up with stationary cursor.
<CtrlD>	Scroll window down with stationary cursor.
<CtrlT>	Cursor to top of window.
<CtrlB>	Cursor to bottom of window.
<CtrlE>	Cursor to the top of currently defined block.
<CtrlC>	Cursor to the bottom of the currently defined block.

MOUSE SUPPORT

Multi-Edit has built in mouse support. The mouse may be used in the same manner as the cursor keys: To move the editing cursor, or a menu selection bar. Pressing the left mouse button while in Text-Edit mode will bring up the main menu. When in a menu, the left mouse button may be used to make the menu selection. The right mouse usually cuts out like the <ESC> key. NOTE: The mouse is reconfigurable just like the keyboard. See **KEY MAPPING** under **INSTALL AND SETUP OPTIONS** for more information.

POSITION MARKING

Each time MARK POSITION is invoked the current cursor position is placed onto the top of the Position Marker Stack. The Position Marker Stack stores up to 10 positions. After 10 the oldest position (number 1) will be lost. RETURN TO MARK will return the cursor to the most recently marked position, removing that mark from the list.

MARK POSITION

Default key <F5> or
MENU, Edit, Cursor, Mark

RETURN TO MARK

Default is <ShftF5> or
MENU, Edit, Cursor, Return

5. INSTALL AND SETUP OPTIONS

Multi-Edit is completely reconfigurable from within itself. There is no separate setup program. The Installation and Setup menu allows you to change many different settings such as key mappings, how Multi-Edit responds to different types of files, screen colors, etc. You may optionally save these settings, making them the start-up defaults.

SETUP OPTIONS

VIDEO MODE

Toggles the video mode from 80x25 to an alternate mode, depending on the video adapter in use.

CGA and IBM Monochrome	80x25
EGA	80x25 and 80x43
Sigma Color 400	80x25 and 80x50
VGA	80x25 and 80x50

LOGO SCREEN

Toggling this setting to OFF will prevent the start-up logo screen from coming up when you invoke Multi-Edit.

EXPLOSIONS

If you are not particularly fond of Multi-Edit's "exploding" windows, you can turn them off.

CTRL HELP BOX

When the <Ctrl> key is pressed it changes functions on the key pad to the right of the keyboard. A pop-up window displaying these features appears on the screen when this key is pressed. If this help is not desired set this flag to off to prevent the display of this window.

RESTORE

Toggles restore on and off. If restore is on, then whenever Multi-Edit is exited the status of Multi-Edit, including all files loaded and the window sizes, etc., will be saved in a file called STATUS.ME. This file is specific to the current directory. When Multi-Edit is restarted from that directory, and no files are loaded from the command line, then the previous status of Multi-Edit is restored.

USE EMS MEMORY

Multi-Edit normaly makes use of Expanded Memory, if available. If you experience problems with resident programs that also use EMS memory, you may need to disable Multi-Edit's use of EMS. NOTE: You must exit Multi-Edit and restart for this switch to take effect (this assumes that you select 'Save settings' when you exit from the **Install** menu).

THE EDIT SETTINGS MENU

BACKUPS

Toggles automatic backups on or off (See LOADING AND SAVING FILES).

PAGE BREAK

Prompts for the programmable page break string (See PAGE BREAKS).

TRUNCATE SPACES

Toggles truncation of extra spaces at the end of a line.

AUTOSAVE

Prompts for the number of seconds the keyboard or other function remains idle before saving files. Inputting 0 turns autosave off (See autosave in Loading and Saving Files).

DEFAULT MODE

Toggles the default Insert/Overwrite mode.

TAB EXPANSION

Sets the tab expansion method to Spaces or Tabs with virtual space. See TABS for more information.

INSERT CURSOR

Programs the appearance of the cursor for the insert mode. Choices are Underline, 1/3 block, 2/3 block, and full block.

OVERWRITE CURSOR

Same as Insert Cursor (see above).

WORD DELIMITS

The word delimiters that effect word left and word right may be modified.

SEARCH SWITCHES

Sets the default search switches. See **SEARCH AND REPLACE** for more information.

FORMAT CODE DELIMIT

Sets the format code delimit to be used by the print formatter. See **PRINT FORMATTER** for more information.

MAX UNDO COUNT

Sets the maximum number of changes that will be stored in the undo buffer of each file. The number may be from 0 to 64767.

THE SCREEN SETUP MENU

The Multi-Edit screen is very configurable. The status line, function key labels, and bottom window border may be selectively turned off, leaving more room on the screen for text. Two screen layouts may be created, one of which is selected as the default screen layout. The two layouts may be instantly toggled between with the XZOOM macro, which is assigned to <AltZ> by default.

THE COLOR SELECTION MENU

The color selection menu allows the user to change the Multi-Edit screen colors, including the box shadow color and the character used to draw the box shadows.

KEY MAPPING

The Key Assignment window allows the keyboard to be completely recapped. Most operations are assigned to function keys. The Arrow keys and the PgUp and PgDn keys are used to move through the list.

MULTIPLE ASSIGNMENTS

Editing functions may be assigned to zero, one or two keys, for example <AltF5> and <AltA>. Any key assigned to more than one function or macro will use the last assignment.

FUNCTION KEY LABELS

The labels for the Function keys (<F1>-<F10> and their Shift, Ctrl and Alt combinations) may be redefined.

MOUSE CONTROLS

References to Btn1, MsRt, MsUp, etc refer to using a Mouse, as in Button #1, moving the mouse right, moving the mouse up, etc. These are not keys on the keyboard.

USER MACRO ASSIGNMENTS

A macro can be written and assigned to a key as well. Once a macro is written it may be inserted into the list. See MACRO LANGUAGE REFERENCE for help in writing macros.

THE MOUSE MENU

ENABLING OR DISABLING THE MOUSE

This installation flag enables or disables the mouse. It is sometimes necessary to disable the mouse when running Multi-Edit under programs like Microsoft Windows or Desq-View.

SETTING SENSITIVITY

The variations in sensitivity of mice from one manufacturer to another may require the user to change the Horizontal or Vertical setting in the menu from its default. Experimentation is suggested to create the desired "feel". The higher the sensitivity value, the LESS sensitive the mouse will be.

FILENAME EXTENSION SETUP

Upon loading a file, Multi-Edit checks a list of filename extensions for the extension of the file being loaded. If found then Multi-Edit sets up a format line and settings for indent style, edit mode, wordwrap, language and compiler types, etc. If there is not a specific setup for the filename extension, then the Default selection is used.

The **Filename extensions** selection in the **Install** menu will bring up a list of the currently defined extensions. The options to **Add** a new extension, **Delete** an extension, and to **Modify** an existing extension setup are also provided.

Because of the importance of the **extension specific setup** concept to Multi-Edit's operation, an entire chapter, **FILENAME EXTENSION SPECIFIC SETUP**, has been devoted to it.

THE PRINTER TYPE MENU

The printer type menu allows you to select, create, delete and modify printer drivers for just about any printer.

MODIFYING A PRINTER DRIVER

Each printer driver consists of a list of descriptions, printer codes and the associated print formatter code (see **PRINT FORMATTER**). All may be modified by the user. The printer code itself contains the actual ASCII codes need to make the printer perform the desired function. The printer modify menu displays 3 user editable fields for each of 19 possible printer codes

SPECIAL ASCII CODES

Often codes such as an ESCAPE character or a CARRIAGE RETURN are needed. Since these characters are used by Multi-Edit to abort or accept input, a special method of inputting any ASCII character has been devised. Simply type in a '|' (bar) character followed by the decimal ASCII code. For example, '|27P' would be translate to **ESCAPE P.**

6. COMMAND LINE OPTIONS

From the DOS command line, both filenames (to be loaded) and option switches may be specified in the form *ME <filename list> <options>*. Multiple filenames may be specified, and the DOS "?" and "*" wildcard characters may be used.

The following example would load in the file "readme.doc" and all files in the current directory with a ".SRC" extension. Multi-Edit would be setup for black and white colors, no screen snow, and an enhanced keyboard.

```
ME readme.doc *.src /v /b /e
```

The following parameters may be used when invoking Multi-Edit from the DOS command line. NOTE: The "/" may be replaced with a "-". Also, the switches are NOT case sensitive.

/?	Lists all the command line options to the screen without loading Multi-Edit.
/LY	Enable logo screen., /LN Disable logo screen.
/MY	Enable Mouse, /MN Disable Mouse.
/B	Use Black and White colors only - Important for monochrome monitors.
/V	Use with CGA systems that do NOT have problems with screen "snow" (like portables with LCD screens).
/Xn /Yn	Overrides the automatic detection of screen width (/X) and length (/Y), and sets the screen width or length to n. Use to configure Multi-Edit to run with non-standard video modes.
/O	There is sometimes a problem with Multi-Edit stealing too much time under OS/2. The /O switch will take care of this problem.

/E	An enhanced keyboard is being used. Allows access to the <F11>, <F12> and other enhanced keyboard keys. **DO NOT** use this switch unless you have a system with an enhanced keyboard BIOS.
/An	Leaves n K of memory space of DOS (32k default). If you are using EMS, or are not going to run any DOS command from within Multi-Edit, and you need some more work space, then use /A0.
/R filename	Loads and runs user Macro file - no extension required. See Macro Language Interface - Automatic Macro Loading.
/NR	Prevents restore from occuring.
/KN	Disables changing the key repeat speed. /KY enables changing key repeat speed.

Many of these options are also settable from the **Install** menu, inside Multi-Edit.

DEFAULT COMMAND LINE PARAMETERS

You may setup default command line switches via the DOS environment variable 'ME_DFLT='. We recommend putting 'SET ME_DFLT=*parameters*' into your AUTOEXEC.BAT file.

7. LOADING AND SAVING FILES

LOADING FILES AT MULTI-EDIT INVOCATION FROM THE DOS COMMAND LINE

Filenames may be specified at the DOS command line. The DOS "?" and "*" wildcard characters may be used in the filenames. See COMMAND LINE OPTIONS for more information.

EDIT NEW FILE

MENU, File, Edit-new-file.

Prompts for a file name, creates a fullscreen window, and then loads the file into the new window.

Hitting the <F2> key from the prompt will bring up the DOS Directory Shell, allowing you to select a file and then hit the enter key (see DOS DIRECTORY SHELL).

LOADING A FILE

Default key <ShftF3> or MENU, File, Load.

Prompts for a file name and then loads that file into the current window.

Hitting the <F2> key from the prompt will bring up the DOS Directory Shell, allowing you to select a file and then hit the enter key (see DOS DIRECTORY SHELL).

SAVE A FILE

Default key <F3> or MENU, File, Save.

Prompts for a file name (uses current file name by default), and then saves the file to disk. If BACKUPS (see Installation and Setup) are on, then a backup of the file (if it already exists) is made the first time the file is saved. The backup file will have the extension .BAK.

SAVE A BLOCK OF TEXT

Default key <CtrlF3> or MENU, File, Block-save

Prompts for a file name, and then saves the currently marked block to disk. This can then be printed in the background from the DOS shell.

LOAD A BLOCK OF TEXT

Prompts for a file name, and then loads a file into the current file at the current line.

CHANGING A FILENAME

MENU, File, Name.

The name of the current file may be changed with this function. NOTE: No file saving takes place, and the name of the file is not changed on disk. This is essentially used to create a new file from an existing file.

AUTOMATIC BACKUPS

Multi-Edit normally creates a backup of a file (as it existed when it was loaded) when the user first SAVEs an edited version of the file. This backup is done ONLY on that first save, to maintain a copy of the original file without any of the subsequent editing steps. The filename used for the backup is the same as the original, but with a ".BAK" extension.

Automatic backups may be disabled (see INSTALLATION AND SETUP OPTIONS).

8. INSERTING TEXT

To enter text into a file, simply move the cursor to the desired position and start typing. If Multi-Edit is in Insert mode then the text will be inserted into the file, pushing over any text to the right of the cursor. If Multi-Edit is in Overwrite mode, then existing text will be overwritten by the new text.

All text insertions may be reversed with UNDO (<CtrlF7> by default).

9. DELETING TEXT

The following is a list of the available text deletion functions, and their default key assignments. NOTE: All text deletions may be undone.

<DEL>	Delete character at cursor
<BS> (backspace)	Delete character to left of cursor
<ShftDEL>	Delete word to right of cursor
<CtrlBS>	Delete word to left of cursor
<AltD>	Delete from cursor to end of line
<ShftF8>	Delete line

In addition, blocks may be deleted (see BLOCK OPERATIONS).

10. TABS

Multi-Edit deals with tabs in two ways. The first is to insert an appropriate number of spaces into the text in place of the tab character. The second is two actually insert a tab character and then fill the remaining space with virtual characters (ASCII 255), which will not actually be saved with the text. The method of tab expansion is selected from the **Install-Edit Settings** menus.

The advantage to not expanding tab characters to spaces is that your file sizes will be smaller. Keep in mind, though, that if you use any tab stop settings other that the standard of every 8 characters, problems may arise when you print the file or use the file with other programs, as most will expand tabs to 8 characters.

Whether space characters or virtual space is used, tab expansion follows the tab settings on the format line (see FORMAT LINES).

11. BLOCK OPERATIONS

Blocks of text within a file may be marked by line, column or stream of text. While in block marking mode, the cursor may move in any direction, up, down, left or right. Once the block is marked you may manipulate it with Copy, Move, Deletion, Indent, or Undent, etc. operations.

TO MARK A BLOCK OF LINES

Move the cursor to the first or last line of the desired block. Select Edit, Block, Line-mark from the main menu, or simply hit the <F7> key. This turns block marking on (notice that the line you are on is now highlighted). Move your cursor to the last line of the block (or first, depending on where you started) and select End-block from the Block menu, or hit <F7> again (notice that the highlighting followed your cursor). The block of lines is now marked.

TO MARK A CONTINUOUS STREAM OF TEXT

Stream of text is similar to line marking except that the block's first line begins not at the left margin but at the cursor. The block marking stops on the last line at the cursor position. This is very useful for moving a sentence in a text file.

TO MARK A RECTANGULAR (COLUMNAR) BLOCK

To mark a rectangular block of text starting and ending at any column, select columnar-mark from the Block menu, or hit <ShftF7>. You may now highlight a rectangular section of text. Select Block-end, or hit <F7> or <ShftF7> to end the block. NOTE: This is Insert/Overwrite sensitive.

BLOCK MARKING COMMANDS

LINE-MARK

Default key <F7> or MENU, Edit, Block, Line-mark

Turns on block highlighting for lines, which may be turned off by <F7>.

COLUMNAR-MARK

Default key <ShftF7> or MENU, Edit, Block, columNar-mark

Turns on block highlighting for columnar blocks. which may be turned off by <F7>.

STREAM OF TEXT

Default key <AltF7> or MENU, Edit, Block, Stream

Turns on block highlighting for stream of text which may be turned off by <F7>.

OFF

Default key <CtrlF9> or MENU, Edit, Block, Off

Turns the highlighted block off. NOTE: This is not necessary to mark a new block.

BLOCK MANIPULATION

COPY

Default key <F9> or MENU, Edit, Block, Copy

Copies the currently marked block to the current cursor position. NOTE: Columnar blocks are Insert/Overwrite sensitive.

MOVE

Default key <F10> or MENU, Edit, Block, Move

Moves the currently marked block from it's current position to the current cursor position. NOTE: Columnar blocks are Insert/Overwrite sensitive.

DELETE

Default key <CtrlF10> or MENU, Edit, Block, Delete

Deletes the currently marked block.

INTER-WINDOW COPY

Default key <ShftF9> or MENU, Edit, Block, Window, Copy

Copies a block from another window into the current window at the current cursor position. Inter-window block copies can be reversed using UNDO only if both windows still exit.

INTER-WINDOW MOVE

Default key <ShftF10> or MENU, Edit, Block, Window, Move

Moves a block from another window into the current window at the current cursor position. Inter-window block moves can be reversed using UNDO only if both windows still exit.

BLOCK INDENTING

INDENT

Default key <AltI> or MENU, Edit, Block, Indent

Indents the marked block one tab stop. Applies only to line blocks.

UNDENT

Default key <AltU> or MENU, Edit, Block, Undent

Undents the marked block one tab stop. Applies only to line blocks.

12. UNDO

Default key <CtrlF7> or <CtrlENTER> or MENU, Undo

Undo can, quite simply, undo what you've done. Almost any editing operation can be undone. This includes block operations, text insertions and deletions, major cursor movements, search and replaces, etc.

The only operations that UNDO cannot reverse are DOS shell commands, Disk file operations such as loading and saving, Block moves that exceed memory capabilities, and Window functions such as sizing or window deletion.

The maximum number of operations that the undo buffer for each file will hold is settable from the **Install-Edit Settings** menu, and may be from 0 to 65767.

13. *WINDOWS*

Multiple windows may be opened and used to edit different files, or linked together to edit different sections of the same file or any combination thereof.

The following operations are available from the **Window** menu:

CREATE

Creates a new full screen window.

DELETE

Deletes the currently active window. **NOTE:** All data is lost and cannot be recovered by UNDO as this deletes the UNDO buffer for that window also. A prompt requires confirmation if anything in the window has been modified since the last time the file was saved. Windows may also be deleted from the SWITCH WINDOW menu.

ERASE

Erases the file contained in the current window from memory (not disk). This operation may erase data and cannot be reversed from UNDO. A prompt requires confirmation if anything in the window has been modified since the last time it was saved.

MODIFY WINDOW SIZE

Allows the cursor keys (or mouse) to be used to change the size and position of the current window. This operation does not go into the UNDO buffer.

ZOOM

Toggles the window dimensions between taking up the entire available screen area and its defined shape and size. Also available with <CtrlF5>. If window was defined as full screen, then zoom has no effect.

LINK

Allows the current window to view and edit the same file that is being viewed by another window. A menu of window names will be displayed. Select the window that is editing the file you wish to Link to and then press Return. **NOTE:** If you link two windows together that both have data in them, the data in the current window will be lost. A prompt requiring confirmation occurs if these conditions are present.

UNLINK

If the current window is Linked to another window it will be made independent again. This window will now be independent and empty.

V-SPLIT

Automatically splits the current window vertically into two windows.

H-SPLIT

Automatically splits the current window horizontally into two windows. This function is identical to V-split other than orientation.

SWITCHING WINDOWS

Simple changing of windows is done by shifting to the previous window (<CtrlF1>) or the next window (<ShftF1>) in order of alphabetical sequence of window label. For more powerful window changing use the SWITCH WINDOWS menu.

THE SWITCH WINDOW MENU

Default key <AltF1> or MENU, Window, Switch

The switch window menu provides a list of all windows now in Multi-Edit. From this menu list you can switch windows, preview windows before you switch, hide windows from the Next window function, look at path, window attributes, modification status of a window, as well as save, and delete windows.

This same window is used to do inter-window block operations, and window linking.

SWITCH WINDOW MENU COMMANDS

DELETE

<F2>

Any window may be deleted from the window list. If the window has been modified, the status of which is shown by an '*' at the right of the file name, a prompt will occur to verify the action.

SAVE

<F3>

Any window can be saved to disk using the path which is shown to the left of the file name. If the file already exists in that directory, a prompt will occur for overwrite verification (only on the first save).

HIDE

<CtrlH> or <F4>

Any window may be hidden from view. If a window is hidden, then it will be skipped over by the NEXT WINDOW and LAST WINDOW functions.

HIDE ALL

<AltH> or <F5>

Hides or unhides all of the listed windows. This function works to allow the user to toggle or walk from window to window in a more flexible manner.

VIEW

<AltH> or <F5>

Previews a window before actually making it the current window. Use the <UP> and <DOWN> arrow keys to switch windows and the <PgUp>, <PgDn>, <CtrlHome>, and <CtrlEnd> keys to browse the file. The file becomes the active window by pressing <Enter>. Pressing <Esc> will return you to your starting window.

POS(Substring, String)

RETURNS:	An integer containing the position of substring in String. Returns 0 if substring is not found. NOTE that use of the function XPOS is recommended instead. POS has been retained for compatibility with previous versions.
EXAMPLE:	What_Pos := Pos('WHAT', Test_String);

XPOS(Substring, String, Start_position)

RETURNS:	An integer containing the position of substring in String. Returns 0 if substring is not found. The search will start at Start_position in the string. NOTE that XPOS is both faster and more flexible than POS.
EXAMPLE:	What_Pos := XPos('WHAT', Test_String, 12);

STR_DEL(String, Pos, Num)

Returns:	The result of deleting a substring containing Num characters from String starting at position Pos.
EXAMPLE:	Test_String := STR_DEL(Test_String, 9, 4);

STR_INS(Target_string, Destination_string, Location)

RETURNS: The result of inserting the Target_string into the Destination_string at the specified Location.

EXAMPLE:

```
   {Insert 'HELP' into Test_String at position 0}
Test_Str := STR_INS('HELP',Test_String,9);
```

STR(Integer_value)

RETURNS: A string representation of the integer value.

EXAMPLE:

```
   {Put the current line number on the message line}
Make_Message('Processing line: ' + Str(C_Line));
```

VAL(Result, String)

ACTION: Converts a the string expression String to an integer value, stored in Result.

RETURNS: 0 if no errors are detected, otherwise the position in the string of the error is returned.

EXAMPLE:

```
IF Val(Temp_Value, '1000') = 0 Then
   New_Value := Old_Value + Temp_Value;
END;
```

REMOVE_SPACE(String)

RETURNS: The string, with any preceding and trailing spaces removed, and with any occurrences of more than one consecutive space removed.

GET_EXTENSION(Filename)

RETURNS: A string containing the DOS extension part of Filename.

EXAMPLE:

```
Extension := Get_Extension(Old_file);
```

GET_PATH(Filename)

RETURNS:	A string containing the DOS path (drive and directory) part of Filename string.

EXAMPLE:	If Get_Path(File_Name) <> 'C:\TEST\' Then
	Enter_Dos_Shell;
	End;

TRUNCATE_EXTENSION(Filename)

RETURNS:	A string containing the filename without an extension.
EXAMPLE:	New_File := Truncate_Extension(File_Name) + '.NEW';

TRUNCATE_PATH(Filename)

RETURNS:	A string containing the filename without a path.
EXAMPLE:	New_File := 'B:\NEW\' + Truncate_Path(File_Name);

19. TEXT MANIPULATION OPERATIONS

CUR_CHAR

RETURNS: The character that the cursor is sitting on in the currently active window. If the cursor is sitting on virtual space or beyond the end of a line then ASCII 255 will be returned.

EXAMPLE:

```
Temp_String := Temp_String + Cur_Char;
```

GET_WORD(Delimit_string)

RETURNS: A portion of the current line starting at the current cursor position and ending before the occurrence of any character in the Delimit string, or the end of the line. Moves the cursor to the end of the returned string.

EXAMPLE:

```
  {Get a word}
New_Word := Get_Word( ' $*().!{}' );
```

GET_LINE

RETURNS: A copy of the entire line the cursor is sitting on.

EXAMPLE:

```
  {Get the current line and put it in Temp_Str}
Temp_Str := Get_Line;
```

PUT_LINE(String)

ACTION: Replaces the line the cursor is sitting on with the string.

EXAMPLE:

```
  {Replace current line with new string}
Put_Line( 'This is a new line.' );
```

CR

ACTION:	Performs a carriage return. Will break a line if Insert mode is on. The cursor will be put at the current indent level.

EXAMPLE:

```
   {Insert a new line}
Insert_Mode := TRUE; {Set the insert mode}
EOL;   {goto the end of the line}
CR;    {perform the carriage return}
```

TEXT(String)

ACTION:	Inserts (or overwrites) a string of text into the current line at the current cursor position. This operation is sensitive to the insert/overwrite mode.

EXAMPLE:

```
   {Insert some text}
Insert_Mode := TRUE;
Text( 'This is some text.' );
```

DEL_CHAR

ACTION:	Deletes the current character.

DEL_CHARS(Count)

ACTION:	Deletes Count characters at the current cursor position.

EXAMPLE:

```
Del_Chars( 10 );        {Delete the next 10 chars}
```

DEL_LINE

ACTION: Deletes the current line

EXAMPLE:

```
  {Delete all lines to the end of the file}
WHILE NOT( At_Eof ) DO
  Del_Line;
END;
```

BACK_SPACE

ACTION: Moves the cursor to the left. If Insert_Mode is true, then the current character is deleted, or if the cursor is at column 1, then the current line will be concatenated to the previous line. If Insert_Mode is false (i.e. Overwrite mode) then the character is replaced by a space, and no concatenation of lines will occur.

EXAMPLE:

```
  {Attach current line to previous line}
Insert_Mode := TRUE;
Goto_Col( 1 );          {Goto first character}
Back_Space;             {Perform back space}
```

20. CURSOR POSITION OPERATIONS

HOME

ACTION: Moves the cursor to the currently set indent level.

EOL

ACTION: Moves the cursor to the character position after the end of the current line.

TOF

ACTION: Moves the cursor to the top of the file.

EOF

ACTION: Moves the cursor to the end of the file.

UP

ACTION: Moves the cursor up one line.

DOWN

ACTION: Moves the cursor down one line.

LEFT

ACTION: Moves the cursor to the left one character. If the cursor was at column 1, then the cursor wraps to the end of the previous line.

RIGHT

ACTION:	Moves the cursor to the right one character. **NOTE:** The cursor does NOT wrap at the end of the line. The cursor WILL wrap after column 254 is reached.

WORD_LEFT

ACTION:	Moves the cursor to the left one word. Will wrap to the previous line. NOTE that the WORD_DELIMITS variable may be used to change the delimiters used to define a word.

WORD_RIGHT

ACTION:	Moves the cursor to the right one word. Will wrap to the next line. NOTE that the WORD_DELIMITS variable may be used to change the delimiters used to define a word.

FIRST_WORD

ACTION:	Moves the cursor to the first non-white-space character on the line.

MARK_POS

ACTION:	Pushes the current cursor position onto the position marker stack for the current window.

EXAMPLE:

```
Mark_Pos;
IF NOT( Search_Fwd( The_Word, 0 ) ) THEN
   Goto_Mark;
END;
```

GOTO_MARK

ACTION:	Restores the cursor position to the last position marked and removes that position from the position marker stack.

POP_MARK

ACTION:	Pulls a mark of the marker stack without actually affecting the cursor.

GOTO_LINE(Line_num)

ACTION:	Moves the cursor to the specified line number. Line_num must be between 1 and 32767.
EXAMPLE:	Goto_Line(C_Line + 10);

GOTO_COL(Column_pos)

ACTION:	Moves the cursor to specified column position. Column_pos must be between 1 and 254.
EXAMPLE:	Goto_Col(10);

PAGE_UP

ACTION:	Moves the cursor one window length up.

PAGE_DOWN

ACTION:	Moves the cursor one window length down.

AT_EOF

RETURNS:	True (1) if the cursor is at or beyond the end of the file in the current window.

AT_EOL

RETURNS:	True (1) if the cursor is at or beyond the end of the line in the current window.

C_COL

RETURNS:	An integer containing the current cursor column position the cursor is on.

EXAMPLE:

```
WHILE C_COL <> 10 DO
   Right;
END;
```

C_LINE

RETURNS:	An integer containing the line number the cursor is on in the current file.

EXAMPLE:

```
IF C_Line = 10 THEN
   Down;          {Skip line 10}
END;
```

C_ROW

RETURNS:	An integer containing the current row in the window the cursor is on. 1 = the top row currently visible in the window.

EXAMPLE:

```
   {Move cursor to top of window}
WHILE C_Row > 1 DO
   Up;
END;
```

NEXT_PAGE_BREAK

ACTION:	Moves the cursor to the line after the next occurrence of a page break string.

LAST_PAGE_BREAK

ACTION:	Moves the cursor to line after the previous occurrence of a page break string.

21. TABSTOPS AND INDENTING

TAB_RIGHT

ACTION:	Moves the cursor one Tabstop to the right. If Insert_Mode = TRUE, then a tab stop

TAB_LEFT

ACTION:	Moves the cursor one Tabstop to the left. No characters are inserted or deleted.

INDENT

ACTION:	Performs a TAB_RIGHT, then sets the indent level to the current cursor position.

UNDENT

ACTION:	Moves the cursor one Tabstop back and sets the indent level to the new position. No characters are inserted or deleted.

INDENT_LEVEL

RETURNS:	The column position of the indent level for the current file. This is the position where the cursor will be taken to when a carriage return or a HOME is performed.

SET_INDENT_LEVEL

ACTION:	Sets the indent level for the current file to the current column position.

CRUNCH_TABS(string_variable)

ACTION: When GET_LINE is called it returns the line complete with virtual characters (255s). CRUNCH_TABS takes a string variable, and will remove the virtual characters from it. ***String_variable must be*** a user defined string variable.

EXAMPLE:

```
Def_Str( TStr );
TStr := Get_Line;
Crunch_Tabs( TStr );
```

EXPAND_TABS(string_variable)

ACTION: Expands the tabs in the passed string variable to spaces (if the system variable TAB_EXPAND is FALSE) or virtual characters (if TAB_EXPAND is true). *String_variable* must be a user defined variable.

EXAMPLE:

```
Def_Str( Tstr );
Tstr := '|9|9Test paragraph';
Tab_Expand( TStr );
```

TABS_TO_SPACES(string_variable)

ACTION: Converts all tabs and virtual characters in the string variable to space characters.

EXAMPLE:

```
Def_Str( Tstr );
Tstr := Get_Line;
Tabs_To_Spaces( Tstr );
```

22. BLOCK OPERATIONS

BLOCK_BEGIN

ACTION: Marks the start of a block of lines at the cursor.

COL_BLOCK_BEGIN

ACTION: Marks the start of a columnar block of text at the cursor.

STR_BLOCK_BEGIN

ACTION: Marks the start of a stream of text block at the cursor.

BLOCK_END

ACTION: Marks the end of a block of text, either line or column oriented, at the cursor.

COPY_BLOCK

ACTION: Copies the currently marked block of text to the current cursor position.

MOVE_BLOCK

ACTION: Moves the currently marked block of text to the current cursor position.

WINDOW_COPY(window_num)

ACTION:	Copies the currently marked block of text from the specified window into the current window at the current cursor position.

WINDOW_MOVE(window_num)

ACTION:	Moves the currently marked block of text from the specified window to the current window at the current cursor position.

DELETE_BLOCK

ACTION:	Deletes the currently marked block of text.

BLOCK_STAT

RETURNS:	0 if no block is marked. 1 if a line block is marked. 2 if a columnar block is marked. 3 if a stream of text block is marked

BLOCK_LINE1

RETURNS:	The first marked line number.

BLOCK_LINE2

RETURNS:	The second marked line number.

BLOCK_COL1

RETURNS:	The first marked column number.

BLOCK_COL2

RETURNS:	The second marked column number. Will be > 255 if the block is not columnar.

MARKING

RETURNS:	TRUE if block marking is on. Else FALSE.

BLOCK_OFF

ACTION:	Turns the current block off.

23. SCREEN OPERATIONS

PUT_BOX(UL_Col, UL_Row, LR_Col, LR_Row, BG_Color, FG_Color, Title_string, Shadow)

ACTION: Puts a box on the screen of the dimensions specified by UL_Col, UL_Row, LR_Col and LR_Row. BG_Color is the background color, and FG_Color is the foreground color. Title_string specifies a title that will be centered on the top of the box. If Shadow is TRUE, then the box will have a shadow around it. If a shadow is specified, then remember to include the one vertical line and two horizontal spaces that the shadow takes up, in the box dimensions. Note that the section of the screen where the box is drawn is saved onto a stack. It may be restored with the Kill_Box operation.

EXAMPLE:

```
   {Put a box on the screen for error messages}
Put_Box( 5, 4, 62, 12, White, Red, 'ERROR', TRUE );
   {Prompt user}
Write( 'Hit any key to continue...',6,5,White, Red );
   {Wait for key}
Read_Key;
   {Get rid of box}
Kill_Box;
```

KILL_BOX

ACTION: Erases the last created box from the screen and restores the previous screen contents.

EXAMPLE: (see PUT_BOX)

WRITE(String, Col, Row, BG_Color, FG_Color)

ACTION: Writes a string of text onto the screen at the coordinates specified by Col and Row, with background color BG_Color and foreground color FG_Color. NOTE that this operation does not effect the text in a window in any way, it simply writes to the screen.

MAKE_MESSAGE(message_string)

ACTION: Writes a message onto the message line of the screen (line 2).

EXAMPLE:

```
   {Display the current character}
Make_Message('The character is: ' + Cur_Char);
```

CLR_LINE(Col, Row, Count)

ACTION: Clears a section of a line on the screen starting at the Col/Row coordinate and extending Count spaces. Does not effect the text in a file in any way.

EXAMPLE:

```
   {Clear the message line}
Clr_Line( 1, 2, 80 );
   {Message line may also be cleared with:}
Make_Message( '' );
```

SCREEN_LENGTH

RETURNS: An integer containing the number of character rows (vertical) displayable on the screen.

SCREEN_WIDTH

RETURNS: An integer containing the number of characters (Horizontal) displayable across the screen.

WHEREX

RETURNS: The actual x coordinate of the physical on-screen cursor. This has no relationship to a window's virtual cursor position.

WHEREY

RETURNS: The actual y coordinate of the physical on-screen cursor. This has no relationship to a windows virtual cursor position.

GOTOXY(x, y)

ACTION: Positions the physical cursor at the specified x,y coordinate on the screen. NOTE that this command does not affect the position of a window's virtual cursor.

TOGGLE_VIDEO

ACTION: Toggles the current video mode between 25 line mode and whatever denser mode (if any) is supported by the installed video card (EGA = 43 lines, Color 400 = 50 lines).

VIDEO_CARD

RETURNS:
0 = IBM CGA
1 = IBM Monochrome
2 = MONROE Color
3 = MONROE Monochrome
4 = Sigma Color 400
5 = EGA Color
6 = Hercules Monochrome
7 = VGA Monochrome
8 = VGA Color
9 = EGA Monochrome
10 = Wyse 700

VIDEO_MODE

RETURNS:
0 = 80 x 25 (normal)
1 = 80 x 50 (Sigma Color 400)
2 = 80 x 43 (EGA)
3 = 80 x 50 (VGA)

PUT_LINE_NUM(num)

ACTION: Puts the specified number (num is an integer) into the L[ ] display of the current window.

PUT_COL_NUM(num)

ACTION: Puts the specified number (num is an integer) into the C[ ] display of the current window.

WORKING

ACTION: Invokes the flashing 'WORKING' message in the Multi-Edit status line. The message goes away when REDRAW or NEWSCREEN is called.

SCROLL_BOX_UP(x1, y1, x2, y2, attribute)

ACTION:	Scrolls a region of the screen up 1 line. Attribute is a single byte used to set the colors of the new line. The lower 4 bits are the foreground color, the next 3 bits are the background color and the high bit is the blink/underline status.
EXAMPLE:	Scroll_Box_Up(1,1,80,25, (Back_Color shl 4) or Text_Color);

SCROLL_BOX_DN(x1, y1, x2, y2, attribute)

ACTION:	Same as Scroll_Box_Up, except that it scrolls the screen region down.

CLEAR_SCREEN(attribute_byte)

ACTION:	Clears the screen and sets the colors as defined by attribute_byte, where the lower 4 bits are the foreground color, the next 3 bits are the background color, and the high bit is the blink/underline status.

24. USER INPUT AND INTERFACE

READ_KEY

ACTION:	Suspends execution and waits for a keystroke. Returns the scan codes of the keystroke in KEY1 and KEY2.

EXAMPLE:

```
  {Get keycode}
Read_Key;
  {If key1 = 0, then extended key was hit}
IF Key1 = 0 AND Key2 = 59 THEN
  Make_Message('You hit F1!');
END;
```

CHECK_KEY

RETURNS:	True (1) if a key has been pressed.The scan codes of the keystroke are accessed via the functions KEY1 and KEY2. The equivalent of READ_KEY, without suspending execution.

EXAMPLE:

```
  {If a key has been pressed, then inform user
   of the type of key}
IF Check_Key THEN
  IF Key1 = 0 THEN
    Make_Message('You hit an extended key.');
  ELSE
    Make_Message('You hit a normal key.');
  END:
END;
```

KEY1

RETURNS: An integer containing the first code of the last key that was pressed. A value of zero indicates that an extended key was pressed.

EXAMPLE:

```
  {Check for extended key code}
IF Key1 = 0 THEN
  Key_Code := Key2;
END;
```

KEY2

RETURNS: An integer containing the second code of the last key that was pressed.

EXAMPLE:

```
  {If F1 was pressed, then goto end of macro}
IF Key1 = 0 AND Key2 = 94 Then
  GOTO End_Of_Mac;
END;
```

PUSH_KEY(keycode, scancode)

ACTION: Pushes the passed keycode/scancode combination back onto the Multi-Edit keycode buffer. Up to 16 keycodes may be pushed back.

EXAMPLE:

```
 {Filter out NON extended keys}
Read_Key;
IF (Key1 = 0) THEN
 Push_Key( Key1, Key2 );
END;
```

PASS_KEY(keycode, scancode)

ACTION: Passes the keycode/scancode through to Multi-Edit, which then performs whatever command or macro is assigned to it. This operation is used by the REPEAT macro.

BAR_MENU(Menu_string, Col, Row, Start_num)

ACTION: Creates a horizontal bar menu.

Menu_string is a string constant or variable formatted as follows:
'Item1(h1)Item2(h2)Item3(h3)...'
where Item1 . . Item3 represent menu entries and h1 . . h3 are 2 character indexes for the help file. The first uppercase letter of each item will be highlighted on the screen and used for single key selection (see the example below). Col and Row represent the physical screen coordinates where the menu will be located. Start_num is the number of the menu item that will be highlighted first.

RETURNS: An integer that is the number of the menu item selected, or 0 if the <ESC> key was pressed to exit the menu without making a selection.

EXAMPLE:

```
   {Setup a bar menu}
Result := BAR_MENU('Load(FL)Save(FL)Run(RN)
                                  Compile(CP)',2,2,1);
   {If escape key was pressed, then exit}
IF Result = 0 THEN
   GOTO Exit;
END;
   {If 'Load' was selected, then load the file}
IF Result = 1 THEN
   Load_File('TEST.SRC');
END;
   {If 'Save' was selected, then save the file}
IF Result = 2 THEN
   Save_File;
END; ........
```

V_MENU(String1, String2, String3, Col, Row, Start_num)

ACTION: V_MENU is identical to BAR_MENU (see above) except that the menu is placed on the screen with the items stacked vertically instead of horizontally, and there are three menu strings instead of one. This allows very large vertical menus (as the maximum string length is 254 characters).

STRING_IN(Input string, Prompt string,Length, Col, Row, Help string)

ACTION: Prompts the user to input a string.

Prompt string is a string expression to serve as a prompt to the user. Input string is the string variable where the user response will be stored (one caution is to be sure and initialize Input string either to a default response, or to an empty string). Length is the maximum number of characters the user may input. Help string is the 2 character index into the help file (if <F1> is pressed).

RETURNS: An integer containing 1 (TRUE) if the string was inputted and enter was pressed or 0 (FALSE) if <ESC> was pressed.

EXAMPLE:

```
  {Prompt user for his name}
JX := String_In( Name, 'Enter your name', 20, 2, 2, 'NM');
  {IF escape was pressed then exit}
IF JX = 0 THEN
  GOTO EXIT;
END;
```

25. FUNCTION KEY LABEL OPERATIONS

PUSH_LABELS

ACTION: Pushes a new blank set of function key labels onto the key label stack.

EXAMPLE: (See FLABEL below.)

POP_LABELS

ACTION: Pops the current set of function key labels off the stack, restoring the previous set.

EXAMPLE: (See FLABEL below.)

FLABEL(String, Key_num, Mode)

ACTION: Modifies a key label. String is a six character string variable or literal. Key_num represents the function key number. It is assigned as follows: <F1> through <F10> = 1 through 10 <ShftF1> through <ShftF10> = 11 through 20 <CtrlF1> through <CtrlF10> = 21 through 30 <AltF1> through <AltF10> = 31 through 40 All other numbers are ignored. Mode is the Multi-Edit mode where the label will be used. 0 = EDIT, 1 = DOS_SHELL, 255 = the current labels, regardless of mode.

EXAMPLE:

```
   {Assign label to <F3> from EDIT mode (using the constant
   'EDIT')}
FLABEL( 'SAVE', 3, EDIT );
   {Create new set of labels}
PUSH_LABELS;
   {Assign label to <ShftF5> within current labels}
FLABEL( 'RUN_DS', 15, 255 );
   {Wait for key to be pressed}
READ_KEY;
   {Restore old labels}
POP_LABELS;
```

26. WINDOW OPERATIONS

CREATE_WINDOW

ACTION: Creates a new editing window. If more than 100 windows are created, or Multi-Edit cannot allocate enough memory for a new window, then ERROR_LEVEL will be non-zero. (See ERROR HANDLING)

SWITCH_WINDOW(Win_Num)

ACTION: Switches to the window number denoted by the integer Win_Num. If Win_Num is larger than the number of active windows, then it will wrap to window 1.

EXAMPLE:

```
   {Switch to second editing window}
Switch_Window(2);
```

SIZE_WINDOW(UL_Col, UL_Row, LR_Col, LR_Row)

ACTION: Changes the dimensions of the current window. The dimensions are limited by the current Length and Width of the screen, and will be truncated if those values are exceeded. (See SCREEN_WIDTH and SCREEN_LENGTH)

EXAMPLE:

```
   {Change the dimensions of the current window}
Size window(3,1,23,80);
```

DELETE_WINDOW

ACTION: Completely deletes the current window. NOTE THAT ALL DATA IN THE WINDOW WILL BE LOST!

EXAMPLE:

```
   {If no changes detected in the current
    file then delete the current window}
IF File_Changed = 0 THEN
   Delete_Window;
END;
```

ERASE_WINDOW

ACTION: Erases the contents of the window, but does not delete the window itself. NOTE THAT ALL DATA IN THE WINDOW WILL BE LOST!

EXAMPLE:

```
   {If no changes detected in the current file
    then erase the current window}
IF File_Changed = 0 THEN
   Erase_Window;
END;
```

MODIFY_WINDOW

ACTION: Enters the user-oriented 'modify window' routine. Prompts user for the upper left and lower right corners of the new window size.

LINK_WINDOW(Win_Num)

ACTION: Links the current window to the window identified by Win_Num;

EXAMPLE:

```
  {Link the current window to the next window}
Link_Window( Cur_Window + 1 );
```

UNLINK_WINDOW

ACTION: Unlinks the current window.

LINK_STAT

RETURNS: The number of windows - 1 that the current buffer is linked to. 0 if the buffer is linked to only one buffer.

ZOOM

ACTION: Toggles the current window between it's defined size and full screen size.

REDRAW

ACTION: Updates the display of the current window only.

NEW_SCREEN

ACTION: Redraws the entire screen.

CUR_WINDOW

RETURNS: An integer containing the current window number.

EXAMPLE:

```
  {Switch to next window}
Switch_Window( Cur_Window + 1 );
```

WIN_X1

RETURNS: The upper left hand column of the current window. If it is set to zero, then the left border will not be displayed. The maximum value is limited to the width of the screen minus 2.

WIN_Y1

RETURNS: The upper left hand row of the current window The minimum value is 3. The maximum value is the length of the screen minus 3.

WIN_X2

RETURNS: The lower right hand column of the current window. The minimum value is 4. The maximum value is the width of the screen plus 1 (will cause the right border not to be displayed).

WIN_Y2

RETURNS: The lower right hand row of the current window. The minimum value is 5. The maximum value is the length of the screen minus 1.

WINDOW_COUNT

RETURNS: An integer containing the number of windows currently active.

EXAMPLE:

```
  {Switch to the very last window}
Switch_Window( Window_Count );
```

27. FILE OPERATIONS

LOAD_FILE(Filename)

ACTION: Loads the specified file into the current window. Get error code via the ERROR_LEVEL variable. (See ERROR HANDLING)

EXAMPLE:

```
   {Load the file}
Load_File( ’Test.Txt’ );
   {If error, then run the error macro}
IF Error_Level <> 0 THEN
   Run_Macro( ’ME-ERROR’ );
END;
```

SAVE_FILE

ACTION: Saves the file in the current window. Get error code via the ERROR_LEVEL variable. (See ERROR HANDLING)

EXAMPLE:

```
   {Save the current file}
Save_File;
   {If error, then run the error macro}
IF Error_Level <> 0 THEN
   Run_Macro( ’ME-ERROR’ );
END;
```

FILE_EXISTS(filename)

RETURNS: True (1) if the specified file exists.

EXAMPLE:

```
   {If file exists then load it}
FStr := ’TEST.SRC’;
If File_Exists( FStr ) THEN
   Load_File( Fstr );
END;
   {If error, then run the error macro}
IF Error_Level <> 0 THEN
   Run_Macro( ’ME-ERROR’ );
END;
```

FIRST_SAVE

RETURNS:	True(1) if the current file has been save at least once since the start of the editing session.

EOF_IN_MEM

RETURNS:	True (1) if the end of the file is in memory, false (0) if the EOF is still on disk.

BUFFER_ID

RETURNS:	The ID # of the file buffer that the current window is attached to.

TMP_FILE

RETURNS:	True if a temporary file is being used for this buffer.

TMP_FILE_NAME

RETURNS:	The name of the temporary file being used by this buffer.

FIRST_FILE(filename)

ACTION:	Searches for the first occurrence of the specified filename. The filename may include DOS wildcard characters (* and ?) and a complete path.
RETURNS:	Zero if the file was found, else DOS error code 18 is returned.
EXAMPLE:	(see NEXT_FILE)

NEXT_FILE

ACTION: Searches for the next occurrence of the filename started by FIRST_FILE.

RETURNS: Zero if the file was found, else DOS error code 18 is returned.

EXAMPLE:

```
  {Find all occurrences of a filename string and load them}
TStr := '*.SRC';
IF FIRST_FILE(TSTR) = 0 THEN
   TStr := LAST_FILE_NAME;
   Call Do_Load_File;
   WHILE (NEXT_FILE = 0) DO
      TStr := TStr2 + LAST_FILE_NAME;
            {call a routine to load the file}
      Call Do_Load_File;
   END;
END;
```

LAST_FILE_NAME

RETURNS: The last filename found by a FIRST_FILE or NEXT_FILE search.

EXAMPLE: (See NEXT_FILE)

28. SEARCH AND REPLACE OPERATIONS

NOTE: See the Multi-Edit user's guide for information on regular expressions.

SEARCH_FWD(Search_string, Num_lines)

ACTION:	Searches forward for Search_string from the current cursor position Num_lines number of lines. If Num_Lines = 0, then the search is performed to the end of the file. If Search_string is found, then the cursor will be positioned at the first character of the found item. NOTE that the system variable IGNORE_CASE is used to determine whether or not the search is case sensitive.
RETURNS:	1 if the string was found, 0 if not found.
EXAMPLE:	(See REPLACE)

SEARCH_BWD(Search_string, Num_lines)

ACTION:	Searches backward for Search_string from the current cursor position Num_lines number of lines. If Num_lines = 0, then the search is performed to the top of the file. NOTE that the system variable IGNORE_CASE is used to determine whether or not the search is case sensitive.
RETURNS:	An integer. 1 (TRUE) if the string was found, 0 if not found.

REPLACE(replace_string)

ACTION: The last Search match (see SEARCH) is replaced with replace_string. Does not effect the cursor position. NOTE that the cursor must not have moved since the Search.

EXAMPLE:

```
   {Find all occurrences of 'WHAT' or 'WHERE' and
         replace them with 'WHY', ignoring the case}
Ignore_Case := True;
WHILE Search_Fwd('{WHAT}||{WHERE}', 0 ) DO
   Replace( 'WHY' );
END;
```

29. OPERATING SYSTEM INTERFACE

SHELL_TO_DOS(Cmd_string, Cmd_flag)

ACTION: Shells to DOS and executes the command specified in Cmd_string. If Cmd_flag is true then COMMAND.COM is used to execute Cmd_string, else Mult-Edit attempts to run Cmd_string directly. In order to execute any of the internal dos commands like DIR, DEL or COPY Cmd_flag must be true. If Cmd_string = '' (empty string) then COMMAND.COM is simply executed.

NOTE that unlike Multi-Edit DOS Shell procedures and functions, this procedure may be executed from any Multi-Edit mode.

EXAMPLE:

```
  {Copy a file}
SHELL_TO_DOS( 'COPY ME.HLP A:', TRUE );
```

COMSPEC

RETURNS: A string containing the path to find COMMAND.COM.

EXAMPLE:

```
  {Execute COMMAND.COM}
SHELL_TO_DOS( Comspec, FALSE );
```

GET_ENVIRONMENT(entry_name)

RETURNS: Parses the DOS environment area for entry_name and returns the string associated with it.

EXAMPLE:

```
Tstr := Get_Environment( 'ME_PATH=' );
```

REST_DOS_SCREEN

ACTION: Restores the screen that was saved upon startup of Multi-Edit or upon the last call to SAVE_DOS_SCREEN.

EXAMPLE: (See SAVE_DOS_SCREEN)

14. FILE FORMATTING

MENU, Document

Multi-Edit supports both a normal text editing mode and a document editing mode. In document mode Multi-Edit keeps track of and displays the current page number. The page number is based on the number of page breaks that are before the cursor. The line number displayed in the upper left corner of the window counts the number of lines from the top of the page instead of the top of the file. Multi-Edit's wordwrap and formatting features are available in both modes.

WORDWRAP

When wordwrap is on, a line will be wrapped when the inserting of text causes the line to exceed the current right margin. **NOTE:** Wordwrap assumes one simple rule: A paragraph must end with a blank line or a page break line.

INDENT-STYLE

Selects the current indenting mode (see INDENTING STYLES).

RIGHT MARGIN

Sets the current right margin for wordwrap.

REFORMAT

Reformats from the line the cursor is at to the end of the paragraph. The function works like word wrap. The right margin is set according to the format line. **NOTE:** Reformat also assumes the wordwrap rule: A paragraph must end with a blank line or a page break line.

JUSTIFY

ALSO ASSIGNED BY DEFAULT TO <ALTJ>.
Right justifies, from the current line to the end of the paragraph, by inserting spaces.

UNJUSTIFY

Removes extra spacing (inserted for justification) from the current line to the end of the paragraph (see Justify).

EDIT-FORMATLINE

Enters format line edit mode (see FORMAT LINES). A format line when saved and embedded in the file overrides all other references to the file format such as the filename extension.

SAVE-FORMATLINE

If ON, then the current formatline will be saved with the file (see FORMAT LINES).

FORMAT-CODES

Brings up a menu for inserting codes used by the Print Formatter (See PRINT FORMATTER).

PRINT-FORMATTER

Invokes the Print Format macro (see PRINT FORMATTER).

AUTO-SETUP

Sets up the formatting of the current file according to its filename extension. See FILENAME EXTENSIONS.

DISPLAY TABS

If ON then tab characters will be displayed.

15. FORMAT LINES

The tab stops and right hand margin settings may be modified on a file by file basis via FORMAT LINES. Each file that is loaded into Multi-Edit is given a format line. The format line will be either the default or one set according to the filename extension of the file, both of which are setup via the **Install-Filename Extensions** menu (see 'FILENAME EXTENSION SPECIFIC SETUP').

Most of the time the format lines set by the **Filename Extension Specific Setup** should be sufficient. However should you need to customize the format line for a specific file, the format line may be edited and saved with the file.

A file's format line, if it is not the default format line (in other words it has been modified), is saved with the file as the first line of the file. The code @ME.FORMAT is imbedded in the line. Whenever Multi-Edit loads a file it checks the first line for that code. If found, then that line is used as the format line. **NOTE:** This can be used in source code by putting comment characters at the beginning and end of the format line when you edit it.

Setting Save-formatline to Off from the Document menu will prevent the format line from being saved with the file. Remember, however, that each time the format line is edited the save condition is reset to ON.

A format line when, saved and embedded in the file, overrides the filename extension specific formatting setup.

Selecting EDIT-FORMATLINE from the DOCUMENT menu will cause the current file's format line to be displayed in the first line of the current window. You may now edit the format line as follows:

<TAB>	Inserts (or overwrites) a tab stop into the format line.
R or r	Sets the right margin.
<DEL>	Deletes the character under the cursor.
<BS>	Deletes the character to the left of the cursor.
<END>	Moves the cursor to the end of the format line.
<HOME>	Moves the cursor to the beginning of the format line.
<ENTER>	Exits format-line-edit and saves the changes.
<ESC>	Exits format-line-edit without saving your changes.

Any other text may be entered into the format line, for example, to embed the format line in a comment (for source code). **NOTE:** The cursor will not leave the format line.

16. PAGE BREAKS

Multi-Edit has a page break facility which allows any text file to be broken up and easily paged through just as a full word processor allows you to move between pages. NOTE: This facility is used by but NOT dependent on the document mode feature of Multi-Edit.

The default page break string is a formfeed character (ASCII 12), however it may be changed to any string (up to 20 characters long) from the Installation and Setup menu.

INSERT PAGE BREAK

MENU, Edit, Page-break

Creates a new line above the current line that contains the page break string.

NOTE: A simple <CtrlL> will generate a form feed character.

NEXT PAGE BREAK

Default key <CtrlPGDN> or MENU, Edit, Cursor, Next-page

Moves the cursor down to the line after the next page break.

LAST PAGE BREAK

Default key <CtrlPGUP> or MENU, Edit, Cursor, Last-page

Moves the cursor up to the line after the previous page break.

17. PRINT FORMATTER

MENU, Document, Print-formatter

The print formatter is a macro which formats text files for output to the printer via special character code sequences (see the following pages for a complete list of these codes). Although these codes may be easily entered manually, we have provided a method of automating this process via MENU, Document, format-Codes.

The print formatter does not actually output to the printer, but instead creates a .PRT file in another window, which may be saved and printed in the background via the DOS print spooler (see DOS SHELL).

Some formatting codes generate printer specific output. The printer type is selected from either the PRINT menu or the INSTALLATION AND SETUP menu.

The following pages list the format codes and information on headers and footers, table of contents generator, etc.

The "." delimiter:

> The default format code delimiter is the period character. However the user may define it to be virtually any character via the install menu.

FORMATTING CODES

PAGINATION, HEADERS, AND FOOTERS

Code	Default	Action
.pb n	n=+1	Page break (start page number n)
<FF>		Page break
<FF> 1		Page break, next printed page will be odd
<FF> 2		Page break, next printed page will be even
.df str	Empty	Define footer
.dh str	Empty	Define header
.sf		Suppress footer this page
.sh		Suppress header this page
.hm n	n=2	Set header margin to n
.fm n	n=64	Set footer margin to n

NOTE: <FF> is an ASCII Form Feed character (12).

NOTES ON HEADERS AND FOOTERS

All characters to the right of a .dh or .df are considered to be part of the header or footer. The following commands and variables can be used within header or footer definitions: .ce, .fn, .ti, .da, .pa, and all printer specific codes.

COMMENT AND IGNORED LINES

A comment line (denoted by .np) is not outputted to the .PRT file. Comment lines are useful for defining header and footers, setting margins, etc., without having the line printed as normal text. An Ignored line is outputted to the .PRT file, but it is not processed for formatting codes. This is particularly useful if you need to print a formatting code literally.

LINE SPACING, MARGINS, AND CENTERING

Code	Default	Action
.ls n	n=1	Line spacing is n
.pl n	n=66	Set page length to n
.rm n	n=80	Set right margin to n
.lm n	n=1	Set left margin to n
.tm n	n=5	Set top margin to n
.bm n	n=60	Set bottom margin to n
.ce		Center this line

PRINT ATTRIBUTES

NOTE: These are the default codes and are completely user definable via the PRINT macro.

Code	Action
.u1	Underline On (Printer specific)
.u0	Underline Off (Printer specific)
.e1	Emphasized On (Printer specific)
.e0	Emphasized Off (Printer specific)
.i1	Italic On (Printer specific)
.i0	Italic Off (Printer specific)
.d1	Double Width On (Printer specific)
.d0	Double Width Off (Printer specific)

MISCELLANEOUS

Code	Action
.np	Line will be decoded, but not printed
.ig	Ignore codes; line will not be decoded
.fn	Inserts the current File name
.ti	Inserts the current Time
.da	Inserts the current Date
.pa	Inserts the current Page number

USER DEFINABLE FORMATTING COMMANDS

These are "macro" type formatting commands which the user can define. Since they are processed in a "search and replace" type manner, they can be very versatile. In defining, everything to the right of the command to the end-of-line is considered to be part of the definition. Whenever a matching execute command is encountered, the command is replaced by the definition string. Anything (normal text, other commands, etc.) will be processed as though it were imbedded at the point of the execution command. Legal values for the parameter is 1-10.

.du n	n=1	Define user formatting command n
.ud n	n=1	Execute user formatting command n

EXAMPLE:

```
.du1 This string is too long to type!
.du1 .ul1INSERTING TEXT.dw0
```

Caution: If the insertion of the definition string results in a string length which exceeds 255, the definition string will be truncated until it fits, yielding unpredictable results.

TABLE OF CONTENTS GENERATOR

A table of contents generator is an integral part of the print formatter. It has the ability to generate a table of contents with up to 5 levels, and four different styles or formats. It uses codes similar to the print formatter.

.TC DEFINE TABLE OF CONTENTS.

At least 2 parameters are expected, up to 12 will be recognized.

Parameter 1: The margin to which the page numbers will be right justified.

Parameter 2: 0 = precede page number with spaces, 1 = use periods.

The second and third will determine the amount of indenting for the first level, and the format or style of the first level respectively. Each subsequent pair of parameters will determine the indent and style of subsequent table levels (up to 5 levels). **NOTE:** The indent levels are offsets from the previous indent level, not an absolute character position. Any missing parameters will default as follows:

Right margin = 75
Spaces will be used
Style(for all levels) = 0

Level 1 indent = 5
Level 2 indent = 7
Level 3 indent = 9
Level 4 indent = 11
Level 5 indent = 13

The style parameter can be any integer from 0 to 3 and defines the following possible formats:

0 = plain	Eg: COMPILING A MACRO
1 = numeric	Eg: 1. COMPILING A MACRO
2 = alpha	Eg: A. COMPILING A MACRO
3 = roman numeral	Eg: I. COMPILING A MACRO

.TE - DESIGNATE A TABLE ENTRY

One parameter is expected, which is the table level this entry. Everything on this line beginning with the first word after the parameter, and up to the first occurrence of more than one space, or the delimiters !@#&()[]’“"~\|<>{}*, will be considered part of the table entry.

As an example, if the table definition looked line this:

```
.tc 55 1 4 0 5 1 5 2
```

and there were 3 table entries like this:

```
.te 1 CARING FOR YOUR FERRET
.te 2 FEEDING
.te 3 DRY FOOD
```

The resulting table of contents would look something like this:

```
CARING FOR YOUR FERRET..........................1
        1. FEEDING..............................2
                A. DRY FOOD.....................3
```

18. PRINTING AND PRINTER CONTROL

The following operations may be performed from the Print menu. NOTE: All printer output is sent to the PRN device. The system default is LPT1. Default keystrokes are MENU, Print

PRINT FILE

Prints the file in the current window.

PRINT BLOCK

Prints the currently marked block.

PRINTER SETUP

Invokes a menu to issue printer specific codes for various fonts, etc.

PRINT MARGIN

Sets up a left margin to be added while printing.

PRINTER TYPE

Selects, creates, deletes, or modifies the printer type. Multi-Edit supports a number of different printers. If your printer is not supported, or you wish to modify the settings for a particular printer, then you may select the create or modify entries and enter the desired printer codes. When entering codes, you may use the | character to enter specific ASCII code numbers. See the Macro Language Reference Manual under String Constants.

THE PRINTER MODIFY MENU

The printer modify menu gives the user total control over the printer codes. There are 3 editable fields for each of 19 possible printer codes. The first field is the name of the code. Typical names would be:

10 cpi
Courier landscape

The second field is a 2 character string defining a corresponding print formatter code(the "." delimiter is excluded). This will automatically interface with the print formatter. Any occurrence of this code in a document being processed with the print formatter will be replaced with the corresponding printer control code. The third field is for the printer control code(escape sequence). Consult your printer manual for the proper codes.

NOTE: You can also print a saved file in the background via the DOS shell. After a file of a block is saved to disk simply shell to DOS using either <AltF4> for DOS directory shell and <F8> or <AltF3> for DOS command line shell and use PRINT.COM (see DOS SHELL).

19. SEARCH AND REPLACE

In addition to a standard alphanumeric string search, Multi-Edit has a complete regular expression implementation. See REGULAR EXPRESSIONS for definition and explanation. See also SEARCH OPTIONS.

SEARCH

Default key <F6> or MENU, Edit, Search, Search

Searches file for the inputted string (or expression). This function will find the first occurrence of the inputted string or expression and then stop unless the Prompt switch is set (see 'SEARCH OPTIONS').

SEARCH AND REPLACE

Default key <ShftF6> or MENU, Edit, Search, search-and-Replace

Selectively searches for and replaces the inputted search string with the inputted replace string. **NOTE:** The replace string can be null to effect a search and delete.

NEXT SEARCH

Default key <CtrlF6>

Repeats the last Search or Search-and-Replace operation. This uses the same options (or switches) as the last search operation. Next Search ignores the Global search switch (see 'SEARCH OPTIONS').

ABORTING A SEARCH

<CtrlBREAK>

A Search or Search-and-Replace operation may be aborted with <CtrlBREAK>

SEARCH OPTIONS

The following search options (or switches) are available.

G	Global search (search entire file). Search begins at the top of the file (or end of file if Backward search is on). Cursor is left at the top of file if no match. **NOTE:** UNDO will bring the cursor back to its starting place.
I	Ignore case of alpha characters.
N	No replace prompts.
P	Prompted search. Will continue as long there are any patterns which match.
B	Backwards search.
X	Expressions OFF (turn regular expressions off).
R	Restrict search to lines in the currently marked block. If global search is on, then the search will start at the top or bottom of the block, depending on the state of backwards search.

REGULAR EXPRESSIONS

Regular Expressions provide the ability to deal with boolean expressions, unprintable characters like end-of-line, specific character classes, wild card characters, multiple groups in a single search pattern or replace pattern, etc.

*	match 0..n occurrences of the previous character or expression.
+	match 1..n occurrences of the previous character or expression.
?	Match any character.
[]	Match class of characters. Example: [a-zA-Z0-9_]
[~]	Match any character except the enclosed class.
%	Match beginning of line.
$	Match end of line.
@	Match (or replace) next character literally. Used to search for ?, [,], %, $, {, and }, etc.
{}	Encloses a group of 1 or more expressions. Maximum of 10 groups.
|	Match previous OR next expression. For example, "{BEGIN}|{END}" would match either "BEGIN" or "END". **NOTE:** A character is counted as an expression, therefore "BEGIN|END" would match either "BEGINND" or "BEGIEND".

REPLACE STRING EXPRESSIONS

&	Insert the original found text.
#n	Insert text matched by group number 'n', n = 0 through 9. For example, searching for "{POS(}{?*}{,}{?*}{)}" and replacing with "#0#3#2#1#4" would swap the parameters for the function "POS(str1,str2)".
$	Inserts a carriage return (to break a line).

% Deletes a character. May be used to delete the end of a line, thus concatenating lines.

SEARCH AND REPLACE EXAMPLES

Search = **$|[a-z0-9_][a-z0-9_]** Find next word.

Search = **@$** Find literal $.

Search = **(?*)** Find parenthesis set.

Search = **[=><!@|&]** Find C relational operator.

Search = **{"*"}|{'*'}** Find single or double quote sets.

Search = **[~0-9]** Match any character except a digit.

Search = **{wow}+** Find one or more occurrences of the word 'wow'.

Search = **{if}|{else}** Find 'if' or 'else'.

Search = **{{this is}|{this is not}} a test**
Replace = **#0 a pizza**
Find 'this is a test' or 'this is not a test' and replace with 'this is a pizza' or 'this is not a pizza'.

Search = **%$**
Replace = **%**
Search for a blank line and delete it.

Search = **{;} *{[~]?+;}**
Replace = **#0$#1**
Search for two semi-colon delimited statements on a single line and break the line, removing any preceding spaces on the new line.

20. DOS DIRECTORY SHELL

Default key <AltF4> or MENU, File, Dos-shell.
May also be invoked by MENU, File, Load, <F2>.

The DOS DIRECTORY Shell is used to perform DOS file commands from within Multi-Edit (such as file copying, renaming, directory listings, etc.). **NOTE:** The DOS SHELL functions are invoked from the function keys, not a menu.

DOS DIRECTORY SHELL COMMANDS

The following commands may be invoked from the DOS DIRECTORY SHELL:

MARK/UNMARK FILE

<SPACEBAR>
Marks or Unmarks a file in the directory. The marked file will be indicated by a ">" graphics character.

SELECT FILE, PROGRAM OR DIRECTORY

<Enter>
If the highlighted directory entry is sub-directory then a directory list of it is shown. The current directory is not actually changed. Pressing <Enter> on the '..' directory will display the parent directory.

If the highlighted directory entry is .COM, .EXE, then the program or batch file will be executed.

If the highlighted entry is not a sub-directory or a program/batch file, then Multi-Edit will load the file into a window. If the DOS DIRECTORY SHELL was invoked from a LOAD FILE prompt, then the file will merely be selected, and become the prompt default.

HELP

<F1>
Brings up Help for the DOS Directory Shell.

CHANGE DIRECTORY

<F2>

Changes the working or default subdirectory. The current drive may also be changed with this function.

DELETE FILE

<F3>

Brings up a prompt requesting the file(s) to delete. The highlighted directory entry will be the default.

COPY FILE

<F4>

Copies the file highlighted in the directory display (or you may type in another filename) to the prompted destination. May be a file, drive, directory or device.

RENAME FILE

<F5>

Renames the highlighted file. Prompts for the new filename.

GET DIRECTORY

<F6>

Prompts for a path, then displays a directory listing. Works just like DOS's DIR command.

RUN DOS COMMANDS

<F7>

Shells to dos. See 'RUNNING DOS FROM MULTI-EDIT' for more information.

PRINT FILE

<F8>

This function allows you to print a file using the DOS background print spooler, PRINT.COM. After specifying the file to be printed, you may immediately continue other work within Multi-Edit, or actually exit Multi-Edit without disturbing the PRINT operation.

PLEASE NOTE THE FOLLOWING:

A. PRINT.COM program must either be in the current subdirectory or there must be a path to where the program resides.

B. PRINT.COM will NOT accept a path as part of the filename to print for DOS versions prior to DOS 3.0.

C. PRINT.COM should already be resident for optimal use of memory.

LOAD A FILE

<F9>
Loads the selected file into Multi-Edit. If the DOS shell is being used from a prompt (see Loading and Saving files) this key selects the file.

SORT DIRECTORY

<F10>
Sets the sort key for the directory. The directory listing may be sorted by Name, Extension, Size, Date and Time. A - may be used to reverse the sort order.

DELETE MARKED FILES

<ShftF3>
Deletes all files currently marked. Prompts for verification first.

COPY MARKED FILES

<ShftF4>
Copies all files currently marked to the prompted destination.

PRINT MARKED FILES

<ShftF8>
Prints the marked files via the DOS Print command. See 'PRINT FILE' above for more information.

LOAD MARKED FILES

<ShftF9>
Loads all marked files into windows.

21. RUNNING DOS FROM MULTI-EDIT

Default key <AltF3> or MENU, Support, Dos

Multi-Edit frees up as much memory as possible, and then runs a copy of COMMAND.COM. If Expanded Memory is available, Multi-Edit will swap all of its data space out of conventional memory, freeing up to 460k of RAM for DOS.

Type EXIT to return to Multi-Edit.

22. *KEYSTROKE MACROS*

Default key <Altf10>

Sequences of keystrokes may be recorded and played back from both Text Edit mode and the DOS SHELL. To start recording keystrokes simply hit <AltF10>, and go on editing (or working in the DOS SHELL). An "R" will appear in the upper right corner of the screen when in Record mode. When you have typed in the desired keystroke sequence, hit <AltF10> again. You will then be asked to type the key that you wish to assign the keystroke macro to (or hit <ESC> to discard the new macro). Any key may be used. If the macro is assigned to a key which is already defined to another function, the key will take on the new function.

NOTE: Recording of keystrokes cannot be started from a menu or prompt.

EDITING KEYSTROKE MACROS

Once a recorded keystroke macro has been assigned to a key it may still be edited. The sequence of Menu <F2>, Macro, Save-keystrokes will bring up the Create Key Macro Source menu.

The 'Complete Macro' menu selection creates the Multi-Edit macro language source code for the keystroke macro that has been recorded, whereas the 'Key_In only' selection will just create the 'KEY_IN' statement (see the **Multi-Edit Macro Language Reference Manual** for more information).

Next the mode selection menu will appear. The same key may be assigned to different macros in the **DOS Shell** mode as opposed to the **Edit** mode. Select whichever mode the macro was recorded in.

Now press the key to which the keystroke macro has been assigned. If 'Complete Macro' was selected, then a new window containing macro source code will appear. Additional keystrokes may be recorded in a similar manner and appended to the macro source by selecting Key_in only from the 'Save Keystrokes' operation.

When the macro source file is complete, it must be compiled (default key <CtrlF8>). See MACRO LANGUAGE INTERFACE and THE MULTI-EDIT MACRO LANGUAGE REFERENCE GUIDE for more information on compiling and running macros.

SAVING AND RELOADING KEYSTROKE MACROS

SAVING YOUR MACRO

An error-free compile will automatically save the macro file to disk. It may be named in any manner that DOS allows for file names. NOTE: If two macro files are loaded with the same name, the first one is overwritten.

LOADING YOUR MACRO

The macro may be loaded from the command line, or with the Menu, Macro, Load-a-macro-file sequence. Macros may be auto-loaded from the STARTUP macro. See MACRO LANGUAGE INTERFACE for further details.

23. LINE DRAWING

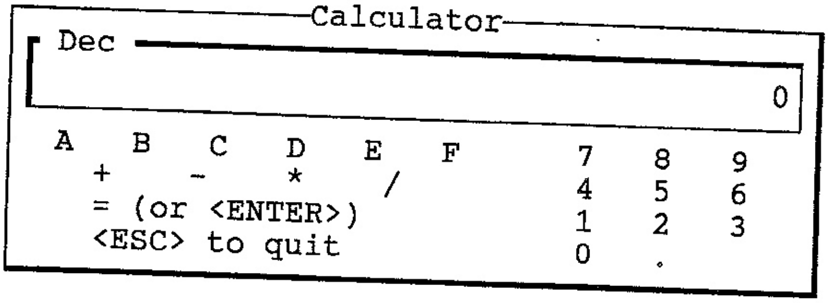
MENU, Support, Line-draw

Allows you to 'draw' with the extended graphic characters. The function will make all proper connections required to connect lines as they intersect with no additional user effort. In order to have a printout of the result your printer must be able to handle the extended ASCII set. To draw in Line-draw:

Arrow Keys	Move the cursor around the screen without drawing any lines.
Shifted Arrow Keys	Draw lines in whatever mode selected.
<F2>	Toggles between single line, double line, and erase line mode.
<ESC>	Exits Line-draw mode.
<PgUp>	Move cursor one window up.
<PgDn>	Move cursor one window down.
<END>	Move cursor to end of line.
<HOME>	Move cursor to start of line.

24. CALCULATOR

Default key <AltF2> or MENU, Support, Calculator



The calculator uses standard algebraic notation and supports Floating point Decimal, Hex and Binary formats. The Hex and Binary functions are accurate to 32 bits. The floating point range is 1E-38 through 1E+38 with a mantissa of 11 significant digits. The following operators and functions are supported:

+	Addition	
-	Subtraction	
*	Multiplication	
/	Division	
<F2>	Full clear	
<F3>	Clear current entry only	
<F4>	Binary AND	
<F5>	Binary OR	
<F6>	Binary XOR	
<F7>	Change BASE. Switches between Decimal, Hex and Binary formats	
<F8>	Change Sign	
<F9>	Memory functions	
	<ENTER>	Load memory with current value
	+	Add current value to memory
	R	Recall memory value

The cursor keys (when NUMLOCK is off) may be used to move the calculator window around the screen.

25. ASCII TABLE

Default key <AltF5> or <AltA> or MENU, Support, Ascii-table

Displays a three page ASCII table, which includes character, decimal and hex columns. The ASCII table displays the entire IBM extended character set. Use the <UP> and <DOWN> keys to move between the pages. Hit <ESC> to exit.

26. CONDENSED DISPLAY

MENU, Support, coNdense or default key <AltF4>

Condenses the current file by displaying only the lines that start at or before the cutoff column (defaults to column 1, but is user definable), or lines that contain a specified search string. Use <F2> to specify the cutoff column, or <F3> to specify a search string. The <UP>, <DOWN>, <PgUp> and <PgDn> keys to move quickly through a file. Hit <ENTER> to exit condensed mode and move the cursor to the selected line, or <ESC> to exit and return the cursor to it's original location.

Condensed display is very useful for editing large source code files (particularly for a structured language), as it allows you to easily locate and move to different functions, procedures, etc.

27. FILENAME EXTENSIONS

MENU, Install, Filename-extensions

Upon loading a file, Multi-Edit checks a list of filename extensions for the extension of the file being loaded. If found then Multi-Edit sets up a format line and settings for indent style, document mode, wordwrap and language and compiler types. You may add extensions to this list, delete extensions from it, and modify the settings for each extension from the Filename extensions selection in the Install menu.

EXAMPLE:

> A .doc file may automatically be set in document mode, have word wrap on, tabs set at 8, and a specific margin. For programming in C however, a .c extension may request text mode, no word wrap, a different margin, a Microsoft compiler, and upon invoking the COMPILE command (default <CtrlF8>) have a BATCH or MAKE file (local to that directory) called UPDATE.BAT execute.

File extension

If you change the filename extension from this menu, what will actually happen is that a copy of this extension setup will be created for the new extension.

Edit mode

Specifies text or document editing mode. For language applications, text would be specified, for documentation files, document mode would probably be desired.

Word wrap

Word wrap is probably only desired for documentation files.

Indent

Determines the indent stye - Off, Auto or Smart. See Indenting Styles for more info.

Tab spacing

Sets the number of spaces to expand tabs and indents to.

Format line

Allows for a specific format line to be assigned to the loaded file. Overrides **Tab spacing** above. When editing the format line, hit <ENTER> to accept your changes, or hit <ESC> to cancel the format line, and go back to using **Tab spacing**. See **Format Lines** for more info.

Right margin

Sets the right margin used for word wrap.

Language type

Selects the language type to be used for smart indenting, brace matching and template editing.

Compiler Command

This is the command line that is to be used to compile the source file. The following identifiers may be inserted into the command line:

<FILE>	inserts the path and filename (but not the extension).
<EXT>	inserts the filename extension.
<NAME>	inserts the filename, but no path or extension.
<PATH>	inserts just the path.
<NR>	prevents Multi-Edit from redirecting the standard output into a file.
<ME_PATH>	inserts the Multi-Edit directory path.

Compiler prompt

If **Compiler prompt** is on then when the Compile command is executed, a prompt will come up with the default compiler command line. This allows the compiler command line to be changed before being executed.

Compiler type

Select the type of compiler you are using. This is needed in order to properly auto-locate compiler errors. Many compilers conform to the MICROSOFT error output standard. If the compiler you are using is not on the list, try MICROSOFT.

Large compile

See LARGE COMPILE for information on the Large Compile option.

Clear Screen

If the compiler appears to be writing through to the screen (i.e. the Multi-Edit screen is getting messed up) then turn this option on.

Save all files

If **Save all files** is on, then all changed files, not just the one being edited, will be saved upon executing **Compile**. This is a safety feature, and it really is a good idea to always have this turned on.

Post-load macro

Specifies the name of a macro to be run after loading the file. This allows even more extensive customization.

See LANGUAGE SPECIFIC FEATURES and INSTALL AND SETUP OPTIONS.

28. INDENTING STYLES

MENU, Document, Indent-style

There are three type of indenting in Multi-Edit: Manual indenting, auto-indenting, and smart auto-indenting. Manual indenting is always available. Auto-indenting and smart auto-indenting are selected from the DOCUMENT menu (see also FILENAME EXTENSIONS).

See BLOCK OPERATIONS for information on indenting and undenting blocks of text.

MANUAL INDENT

Default key <F4>

Tabs the cursor to the next tab marker set in the format line and then sets the left margin at that position. When <ENTER> is pressed the cursor will return to the new left margin instead of column 1. The home key on the keypad will also respond to the indent position.

MANUAL UNDENT

Default key <ShftF4>

Tabs the Cursor back to the previous tab marker set in the format line and then sets the left margin at that position. The home key on the keypad will also respond to the indent position.

AUTO-INDENT

When <ENTER> is pressed while editing a file, the cursor will be left at the indent level of the previous line. Auto-indent may be selected or turned off via MENU, Document, Indent-style. The home key on the keypad will also respond to the indent position.

SMART AUTO-INDENT

See LANGUAGE SPECIFIC SUPPORT.

29. LANGUAGE SPECIFIC FEATURES

SMART AUTO-INDENT

Smart auto-indenting is language specific. When <ENTER> is pressed while editing a file, a macro is executed that backtracks through your source code to determine the proper indent level, and then indents the new line appropriately. Smart auto-indent may be selected or turned off via **MENU**, **Document**, **Indent-style**, and may also be selected in the **Filename Extension Setup** menu.

KEYWORDS FOR PASCAL INDENTING

(these keywords are not case sensitive)

var, type, const, procedure, function, begin, if, while, repeat, with, for

In addition an end statement will cause an undent. If this is not desired the user may comment out this feature in the code of the macro PAS_IND in the macro file PASCAL.SRC.

C INDENTING

The C indenting macro looks at the line-ending characters (ignoring comments, including C++ '//' comments, labels, and white space) for the current and preceding lines, then decides whether and how much to change the indentation from that of the current line. In addition, if the CR was done inside a (C-style, /* */) comment, the "line of asterisks" style of comment continuation is automatically supplied.

AUTOMATIC LINE NUMBERING FOR BASIC

There is not a smart auto-indent for BASIC. There is, however, automatic line numbering, that will create line numbers for your program upon hitting <ENTER> (see FILENAME EXTENSIONS).

BRACE, BRACKET AND STRUCTURE MATCHING

MENU, Support, Match or default key <AltF6>

Currently supports Pascal and C.

For Pascal:
Matches BEGIN/END, CASE/END and parenthesis.

For C:
Matches {} and parenthesis.

Place cursor on item to match. Hit MATCH and the cursor will be taken to the matching brace or statement.

TEMPLATE EDITING

Default key <AltF9> or <AltT> or MENU, Support, Template

With template editing, you can type a single character, hit Template and an entire construct will be create (for the language you are programming in), based on that character.

TEMPLATE EDITING IN C

i	Builds an ’if () {’ construct.
w	Builds a ’while () {’ construct.
f	Builds a ’for () {’ construct.
{	Builds matching braces.
s	Builds a ’switch () {’ construct.
d	Builds a ’do { } while ();’ construct.
/*	Builds a date/time stamped comment block.

TEMPLATE EDITING IN PASCAL

Since pascal keywords are not case sensitive, the case of the templates will be determined by the case of the character you type. NOTE: Since ’FOR’ and ’FUNCTION’ both start with ’F’ you will be prompted for the one to use, unless you type in the second character.

B	Builds a matching ’BEGIN/END’ construct.
I	Builds an ’IF () THEN’ construct.
W	Builds a ’WHILE () DO’ construct.
F	Prompts for’FOR-T’ or ’FUNCTION’.
FU	Builds a ’FUNCTION () : ;’ construct with matching ’BEGIN/END’.
FO	Builds a ’FOR := TO DO’ construct.
P	Builds a ’PROCEDURE ();’ construct with matching ’BEGIN/END’.
C	Builds a ’CASE OF’ construct with matching ’END’.
R	Builds a ’REPEAT UNTIL ();’ construct.

TEMPLATE EDITING IN THE MULTI-EDIT MACRO LANGUAGE

See Multi-Edit Macro Language Reference section for additional information.

$M	Builds a ’$MACRO ; END_MACRO;’ construct.
I	Builds an ’IF () THEN’ construct with matching ’END;’.
W	Builds a ’WHILE () DO’ construct with matching ’END;’.

TEMPLATE EDITING IN BASIC

To be as compatible as possible with interpreted and compiled basic, the following convention is followed: If the line the cursor is sitting on is numbered, then the construct created will include line numbers. Otherwise, the construct will have no line numbers.

I	Builds a ’IF THEN construct.
F	Builds a ’FOR NEXT construct.
W	Builds a ’WHILE WEND construct.

COMPILING FROM MULTI-EDIT

Default key <CtrlF8> or MENU, Support, coMpile

To compile within Multi-Edit a filename extension setup with a compiler type and compiler command must exist for the extension of the file you wish to compile. It is suggested that batch file or Make commands be used, and that a standardized naming scheme be used for batch and make files. See FILENAME EXTENSIONS for more information.

When you invoke the compiler from within Multi-Edit, the command line will automatically be appended with

```
> MEERR!.TMP
```

in order to redirect the output to a file called MEERR!.TMP. This file is then automatically loaded and parsed into a newly created window, and if there are any compiler errors, the cursor will move to the first compiler error in the source code.

NEXT ERROR

Default key <CtrlF2>

Moves the cursor to each successive compiler error. The MEERR!.TMP window may be accessed for additional compile information.

COMPILER COMMAND LINES

A different Compiler Command line may be created for each filename extension setup. The Compiler Command Line allows easy interface to other programs. It is not limited to compilers.

The following codes may be embedded in the compiler command line:

<FILE>	Inserts the path and filename (without the extension) of the file in the current window.
<EXT>	Inserts the extension name of the file in the current window. Example: <FILE>.<EXT>
<NAME>	Inserts the filename, without path or extension, of the file in the current window.
<PATH>	Inserts the path of the file in the current window.
<ME_PATH>	Inserts the path to find the multi-edit files. Includes the ending backslash (i.e. 'C:\ME\').
<NR>	Prevents redirection of the program's standard output.

EXAMPLES:

TPC /Q <FILE>;
Compiles current file under Turbo Pascal 4.0.

WSP <FILE>.<EXT>
Invokes Webster's spelling dictionary to check the current file for spelling errors.

COMPILER PROMPT

Setting the Compiler prompt flag to ON, under Filename Extension Setup ,will bring up a prompt (with a default command line) when the COMPILE command is invoked.

REDIRECTING OUTPUT IN BATCH FILES

When a batch file is used in the compiler command line, the redirection command is ignored by DOS. Because of this the redirection will have to be added into the batch file itself. A typical command line in a batch file (for MICROSOFT C) would be:

```
MSC %1; > MEERR!.TMP
```

Multiple compilation steps (i.e. multiple passes or linking) may be processed from a batch file. To avoid overwriting the MEERR!.TMP file on each step, the DOS append symbol must be used:

```
LINK %1; >> MEERR!.TMP
```

LARGE COMPILE

Sometimes a compiler will require more memory than Multi-Edit can free up (particularly C compilers). The Large Compile option (which is specified for each filename extension) is used for such situations. The Large Compile option requires that Multi-Edit be invoked with the MEC batch file supplied on the program disk. NOTE: If EMS memory is available, then Large Compile will probably not be needed.

When you perform a compile, and Large-Compile is ON, the following will happen:

1. The current status of Multi-Edit will be saved
2. Multi-Edit will be exited.
3. The compiler will be invoked with error redirection.
4. Multi-Edit will be restarted (with the old state restored).
5. The error file will be brought up and parsed.

The MEC.BAT file basically sets up a loop based on Multi-Edits exit code. The file MECOMP!.TMP is created by Multi-Edit before exiting and is used as the input for running COMMAND.COM. The MECOMP!.TMP file consists basically of the name and commandline of the program to be executed.

NOTE: To use Large Compile with some compilers (notably Turbo-C 1.0) you may have to increase the number of files openable on your system to 2 or 3 above the default. This is done via the FILES= command in your CONFIG.SYS file. We recommend a setting of FILES=20.

ALSO: Large Compile will probably NOT be needed if you have Expanded (EMS) memory.

30. MACRO LANGUAGE INTERFACE

RUN MACRO

Default key <F8> or MENU, Macro, Run-a-macro

Prompts for a macro name to execute. The name may be any legal DOS filename, without an extension. Multi-Edit will first look for the macro in memory. If it doesn't find it, then it will search the current directory (if no path name was specified). If it still doesn't find the macro then it will check the Multi-Edit files directory. It then believes it doesn't exist and reports an error.

LOAD A MACRO FILE

MENU, Macro, Load-a-macro-file

Loads a macro file from disk. If no path name is specified then Multi-Edit first checks the current directory, then the Multi-Edit directory.

UNLOAD MACRO

MENU, Macro, Unload-a-macro

Erases a macro from current Multi-Edit memory.

AUTOMATIC MACRO LOADING

When Multi-Edit is invoked, it looks for a macro file called STARTUP.MAC, first in the current directory, and then, if it isn't found, in the Multi-Edit directory. The STARTUP macro may be used for automatic loading of user written macros, and custom initialization sequences.

EXAMPLE:

```
$Macro startup;
        load_macro_file('your_mac');
        load_macro_file('macro2');
        set_global_str( 'my_settings', 'custom' );
End_macro;
```

31. MISCELLANEOUS STUFF

COMMAND REPEAT

Default key <CtrlR>

Command Repeat allows any keystroke (including Function keys, and Ctrl and Alt keys) to be repeated up to 2 billion times. When **Command Repeat** is invoked, a number may be entered to set the repeat count, and the key repeat is pressed. The keypress is then repeated the set number of times. A single UNDO will reverse the entire operation.

TIME AND DATE STAMPING

MENU, Edit, Time/date-stamp or default key <ShftF2>

Inserts the Date and Time into the file at the current cursor position in the format: mm/dd/yy HH:MMam (or pm). NOTE: This is Overwrite/Insert sensitive.

EDITING THE HELP FILE

If the user desires to change the help file this can be accomplished by simply editing ME.HLP. NOTE: The search functions to go from page to page in the HELP are dependant on the characters embedded in the page break string, so use some caution there. The character ^ is used to enable and disable a boldface mode.

APPENDIX A: EDITING FILES LARGER THAN MEMORY

Multi-Edit automatically swaps files, or portions of files, in and out of memory as needed. If a file is initially too large to fit into available memory, then only a portion of that file will be loaded at a time. When memory gets low, Multi-Edit will do its best to free up memory by swapping inactive files, or inactive parts of the current file, out to disk.

If a file that has been modified needs to be swapped to disk, then Multi-Edit will build a swap file where the changed version of the original file will be stored. For DOS versions 3.0 and above, Multi-Edit will use the DOS "Create Unique File" function to build the temporary files. This insures that, even in a network environment, Multi-Edit is not overwriting an existing temporary file. For versions of DOS less then 3.0 (i.e. 2.0, 2.1, 2.11) these swap files are named by the convention "!MEnnnnn.TMP" where "nnnnn" is a number corresponding to the unique Multi-Edit buffer ID of the file. "!ME2.TMP" and "!ME9782.TMP" are both valid swap file names. Swap files are always marked as read-only, to prevent accidental deletion. All swap files are automatically deleted when Multi-Edit is exited.

APPENDIX B: DESQ-VIEW, WINDOWS and OS/2

Multi-Edit is fully compatible with DESQ-VIEW and Microsoft Windows, and it may be run in the OS/2 compatibility box.

Multi-Edit requires that at least 190k of RAM be allocated to it. 256k is the preferred minimum. Multi-Edit will detect the presence of DESQ-VIEW or MS Windows on startup, and appropriately redirect its screen I/O. Under some circumstances, it may be desired for Multi-Edit to write directly to the screen whether DESQ-VIEW/WINDOWS is present or not. The is done with the /D command line switch.

When running in the OS/2 compatibility box, Multi-Edit may steal an unfair percentage of processor time. If this is found to be the case, simply start Multi-Edit with the /O command line switch.

APPENDIX C: RUNNING WITH OLDER VERSIONS OF DOS

Multi-Edit is compatible with DOS versions 2.0 and above. However, with versions of DOS below 3.0 there are some differences in operation.

LOCATING THE MULTI-EDIT PATH

For DOS versions 3.0 and above, Multi-Edit will automatically find the directory where it's .EXE file and other files are located. For DOS versions 2.xx the environment variable "ME_PATH" must be set to the path where the Multi-Edit files are located. This is done with the DOS "SET" command. Example:

```
SET ME_PATH=C:\ME
```

The best approach is to put this command in your AUTOEXEC.BAT file.

MOUSE SUPPORT

If you are running DOS version 2.xx, it is very important that you do NOT enable Multi-Edit's mouse support unless you have a mouse driver loaded in your system. Doing so may lock up your system. The mouse may be disabled with the /MN command line switch.

TEMPORARY FILE NAMES

As noted in APPENDIX A, the way Multi-Edit names temporary files is different. See APPENDIX A for more details.

INDEX

/? (16)
/An (17)
/B (16)
/D (70)
/E (17)
/LY (16)
/MY (16)
/NR (17)
/O (16)
/R (17)
/V (16)
/Xn (16)
/Yn (16)
ASCII
  table (55)
ATTRIBUTES
  print (37)
AUTO-SETUP (32)
AUTOEXEC.BAT (17)
Autosave
  setting (11)
Backups
  creation (19)
  enabling (11)
BASIC (60)
BATCH FILES (66)
Block
  Columnar-mark (23)
  Copy (24)
  Delete (24)
  Indent (25)
  Inter-window (24)
  Line-mark (23)
  marking (22)
  Move (24)
  Off (23)
  Stream (23)
  Undent (25)
Bulletin Board System (1)
C (60-62), (67)
CALCULATOR (54)
CENTERING (37)
Change Directory (48)
Color
  settings (12)
COMMAND LINE
  options (16)
Command Repeat (69)
COMPILING (64)
  COMPILER COMMAND LINES (65)
CONDENSED DISPLAY (55)
Copy
  file (48)
  marked files (49)
Ctrl Help Box (10)
cursor
  keys (7)
  shape (12)
DATE
  Stamping (69)
Delete
  file (48)
  marked files (49)
  text (20)
DESQ-VIEW (70)
disk swapping (70)
DOCUMENT (31)
DOS (50)
DOS DIRECTORY SHELL (47)
EDIT SETTINGS (11)
EMS (11)
Error
  compiler (64)

Explosions (10)
File
  backups (19)
  block (19)
  Copy (48)
  Delete (48)
  Edit New (18)
  filename (19)
  load from directory shell (49)
  load marked files (49)
  Loading (18)
  Rename (48)
  saving (18)
FILENAME EXTENSIONS (56)
  setup (15)
FOOTERS (36)
Format Line
  default (11)
FORMAT LINES (33)
FORMAT-CODES (32)
Formatline
  edit (32)
  save (32), (33)
FORMATTING CODES (36)
Function keys (6)
  default (6)
  labels (13)
HARDWARE REQUIREMENTS (3)
HEADERS (36)
help (5)
  editing help file (69)
Indent
  auto (59)
  block (25)
  Manual (59)
  Smart (59), (60)
  style (31)
  STYLES (59)
INDENT-STYLE (31)
Insert
  mode (11)
Installation (10)
INSTALLING (3)
JUSTIFY (31)
Key
  MACROS (51)
KEY MAPPING (13)
KEYSTROKE MACROS (51)
  EDITING (51)
  SAVING (52)
LANGUAGE SPECIFIC (60)
LANGUAGE.SRC (68)
Large Compile (11), (67)
LINE DRAWING (53)
LINE SPACING (37)
Link (27)
Load
  block (19)
Loading (18)
Logo Screen (10)
MACRO (68)
  Automatic loading (68)
  Load (68)
  Run (68)
  Unload (68)
MARGIN
  right (31)
MARGINS (37)
MATCHING (61)
MEC.BAT (67)
menu
  USING MENUS (5)
Microsoft Windows (70)
mouse (3), (7)
  controls (13)
  Sensitivity (14)
  setup (14)
OS/2 (70)
Page Break
  defining (11)
PAGE BREAKS (34)
PAGINATION (36)
PASCAL (60-62)
POSITION MARKING (9)
Print (48)
  block (41)

    file (41)
    file in background (49)
Print formatter (32), (35)
PRINT.COM (48)
PRINTER (41)
    control (41)
    Pitch (41)
    type (15), (41)
Redirecting (64), (67)
REFORMAT (31)
REGULAR EXPRESSIONS (45)
Rename (48)
Replace (43)
Restore (11)
Save
    block (19)
    file (18)
SEARCH (43)
    Default Switches (12)
    Next Search (43)
    options (44)
    regular expressions (45)
    Replace (43), (45)
Setup (10)
Switch window menu (29)
Tab
    expansion (11)
    stops (33)
TABLE OF CONTENTS GENERATOR (39)
Tabs (21)
TECHNICAL SUPPORT (1)
TEMPLATE EDITING (62)
Text
    deleting (20)
    inserting (20)
TIME
    Stamping (69)
Truncate Spaces (11)
Undent
    Block (25)
Undo (12), (26)
UNJUSTIFY (31)
USER DEFINABLE FORMATTING COMMANDS (38)
Video Mode (10)
WARRANTY (1)
Window
    Create (27)
    Delete (27)
    Erase (27)
    H-Split (28)
    Link (27)
    Modify (27)
    switch (28)
    switch window menu (29)
    Unlink (28)
    V-Split (28)
    Zoom (27)
Windows (27)
Word
    delimits (12)
WORDWRAP (31)
XZOOM (12)
Zoom (27)